Secure Mobile Nodes in Federated Databases

Martin S Olivier
Dept of Computer Science, Rand Afrikaans University, PO Box 524, Auckland Park 2006;
molivier@rkw.rau.ac.za

Abstract

Mobile computers pose special security risks since information contained on them can more easily be compromised. However, availability of information at the location where the mobile computer is to be used often outweighs the disadvantages.

This paper proposes an approach to limit the risks of information compromise by caching only that information on the mobile computer that the user of the mobile computer is authorised to access. Mechanisms to manage transfer of such information are described.

The mechanisms that are proposed are described as extensions to federated databases. In particular, they are intended for federated databases where the security policies of the different sites may differ. The mechanisms that transfer information to and from the mobile node use the different security policies to determine what information can be transferred and how it is transferred.

Keywords: *Distributed Systems: Distributed databases; Database Management: Mobile computing; Security and Protection*
Computing Review Categories: *C2.4; H2.m; K6.5*

1 Introduction

Mobile computing poses special security problems. By definition computers used for mobile computing are often removed from the protective environment of the organisation that owns them. In addition such computers are not linked to their host machines for significant periods—in some cases they are disconnected for the entire time they are physically away from the host; in other cases contact can be made (by cellular phone or other dial-up facility) when required. Online monitoring of activities on the mobile computer is therefore impractical. In addition, mobile computers are easily stolen making prolonged attacks possible, disks can be removed and read on other computers, radio communication can be intercepted and often current operating systems used on mobile computers offer little or no security measures.

It is generally accepted that mobile computers have to *cache* information they require so that it is available—even during disconnected operation. A number of approaches to locally cache (distributed) information and manage the cache during disconnected operation have been proposed and even implemented (see section 2). This paper considers caching of potentially sensitive information on relatively untrusted mobile computers. In particular we propose that different versions of sensitive objects are maintained (or created) for use on mobile computers. The paper also proposes an architecture that handles the transfer of objects between the mobile computer and the 'fixed' database. This architecture addresses issues such as obtaining the correct version of an object to be transferred to the mobile computer or even constructing an appropriate version, obtaining permission from the owner of such information before caching the information and the secure synchronisation of the cache with information retained on the fixed database.

When information owned by different organisations who share a database is to be cached on a mobile computer, additional complications arise. Consider, for example, a sales representative of a wholesaler who requires information from a number of suppliers; the sales representative visits retailers and wants to have relevant information available from a mobile computer. If the wholesaler and the suppliers share a federated database, such information will be easily accessible. However, it is possible that the suppliers hold different views about the sensitivity of the particular information they supply (such as cost price, availability and even manufacturing process). The intention of this paper is to make as much as possible of this information available on the mobile computer, but still satisfy all the security requirements of the suppliers.

Work on multipolicy federated databases is still in its infancy. A multipolicy federated database is a distributed database built to support more than one security policy. The various participants in the federation—that is, the various organisations who own the nodes that form the federated database—are each able to specify the security policy that applies to data owned by it. This policy is then to be honoured by all other members of the federation. A federal security policy—that is a policy that applies to all members of the federation—may also be agreed on by the members of the federation.

In [12] a model has been described that supports a multipolicy federated database. This model assumes

that a *trusted common core* (TCC) exists at all sites of the federation. This TCC is trusted by all federation members and ensures that all local security policies are enforced at other sites. However, in a heterogeneous federation it may not be possible to implement an identical TCC on all nodes. Legacy systems cannot easily be modified; some systems are inherently less trusted in a federation, because of their location, hardware used or other reasons. In particular, if a node is mobile, it may not be trusted for the reasons given earlier. This paper is a first attempt to accommodate such inherently different nodes in a secure federation.

The paper is organised as follows: Section 2 contains a background discussion of mobile computing and security in federated databases. Section 3 describes the operation of the host node that supports the mobile computer, while section 4 describes the security functionality of the mobile computer. This is followed by a discussion and the conclusion.

2 Background

Mobile computing is a practical reality: Popular magazines often devote major sections of the magazine to this topic. Laptop computers are used to carry information and computing power for individuals who have to work away from the office. Portable modems or links with cellular phones enable the user to connect with network services or send and receive faxes when required. However, data management in the mobile environment still requires much work: Dunham and Helal [4] state that "the status of data management in mobile computing is similar to that of distributed data management versus centralized data management in the late 60s. Namely many of the issues are the same, but the solutions are different."

One major difference between mobile computers and other nodes in a distributed database is the fact that the mobile computers cannot indefinitely remain connected to the network: the capacity of batteries used, the cost of cellular connections and the existence of areas not reached by cellular or other radio signals all preclude this. It is therefore necessary to manage data on the mobile unit such that disconnected operation is possible for significant periods.

It is generally accepted that the architecture of a mobile system includes a number of fixed hosts and one or more base stations—all connected with a high-speed network [4, 8]. The mobile computers (intermittently) connect to the base stations—either by high-speed wire when the mobile computer is at the base station, or by low-speed radio link in other cases. A mobile computer that moves from the area (cell) serviced by one base station to that serviced by another base station can *cross* to the new base station to make continuous connected operation possible. See [7, 8] for a discussion of data management research issues for mobile computing.

Security for a mobile computer has to balance two aspects: firstly, the fact that the mobile computer is prone to attacks such as those mentioned in the introduction and, secondly, the fact that mobile computers have to support disconnected operation. The first aspect requires that as little as possible sensitive information is placed on the mobile computer. The second aspect requires that as much as possible of the required information is placed on the mobile computer.

Disconnected operation can be supported by caching information on the mobile computer. Most work on such caching has been done in the context of distributed file systems. Obviously the nature of mobile computers require that such caching has to be on a local disk of the mobile computer. Coda is one example of a distributed file system that uses disk caching and "disk caching is critical to disconnected operation in Coda" [18]. DOC *(Disconnected Operation Cache)* forms another example of a system that caches files at clients, where the clients are "currently existing notebook computers" [6].

Disconnected operation can obviously lead to inconsistencies in the various cached copies of information. A *pessimistic* approach that only allows one copy to be updated will avoid inconsistencies; an *optimistic* approach allows more than one copy to be updated but then conflicts have to be resolved if they occur [3]. The Coda file system uses an optimistic approach and experience has shown that inconsistencies rarely occur [18]. For simplicity, and to focus on the security aspects in databases, we will use a pessimistic approach in this paper. Obviously this limits availability of an object to one mobile user and will have to be addressed in future research. It is substantially harder to implement a distributed database compared to a distributed file system because of the "concurrent read and write sharing of data at fine granularity by large numbers of users, combined with requirements for strict consistency of data and atomicity of groups of operations" [18]. However, since more semantic knowledge is available in a database than in a file system (and in particular in an object-oriented database) automation of the resolution of inconsistencies holds more promise than in the case of distributed file systems.

A mobile computer is often used by a particular individual (and not shared by several users). This is reflected in the use of terms such as *Personal Communication Network*, *Personal Digital Assistants* [7], *Personal Communication Services* and *Personal Information Services and Applications* [5] when considering mobile computing. Our premise is that, since we cannot rely on the mobile computer for complete protection, only information the mobile user is authorised to access will be allowed onto the computer—whether cached or online. The user of the mobile computer is

then expected to handle the computer with the same care as printouts with that information would have been handled. The term *mobile user* will be used to refer to the intended user of the concerned mobile computer.

In this paper we assume that an object-oriented federated database is used, because the reported work forms part of a larger research project—see [12, 13, 15, 14]. We assume that objects have been translated to some internal representation that includes the representation of methods and data—see [15] for details. See [9] for a description of object-oriented databases and [10, 11, 17] for a discussion of security in such databases.

A federated database is a database that provides a relative high degree of site autonomy. See Özsu and Valduriez [16] for a discussion of distributed databases and [16, pp81,89] and [19] for a discussion of federated databases in particular. See [4] for more information on the implications of mobility on distributed databases. See [20] for a discussion of the issues that must be considered when designing a secure federated database.

The federated database in which the mobile computer of the current paper will function is based on the SPO *(self-protecting objects)* model [12]. This model allows the various sites in the federation to maintain different security policies, but still interoperate. In particular, data owned by one site can be used by or relocated to another site and still be as protected at the other site as it has been on its owning site. In the case of a mobile node, this means that information owned by any number of organisations sharing the federated database should be accessible from, or relocatable to the mobile node, without any sacrifice in security.

The SPO model is based on three components: The *trusted common core* (TCC) occurs on all nodes of the federated database and is trusted by all members of the federation. *Trusted extension* (TE) methods are associated with the entities in the database that are to be protected; whenever such an entity is accessed its associated TE methods are activated to perform the required access control. Each site also has a *trusted local extension* (TLE) that supports the TE methods of that site by providing them with information about users, user groups and other information required for security purposes. The TLE and TE methods of a site only have to be trusted by the particular site (and not by other members of the federation). The components of the SPO model are depicted in figure 1. LDB refers to the local database at the site.

See [1] for an alternative approach to support multipolicy federated databases, based on the notion of agreements between sites.

3 The host node

The scenario that has been sketched in the previous sections used a federated database with a mobile node associated with it. For this node a particular site is designated as the host node. For simplicity we assume that the host node is dedicated to supporting the mobile node. (In practice this node can perform additional functions.) Note that the host node does not have to be the same node as the base node: the base node is that node that currently provides the radio connection with the mobile computer; as the mobile computer travels, the associated base node can change; the host node, however, does not necessarily change when the user moves, and communicates with the mobile node via the fixed communication infrastructure and the base node(s).

The mobile node connects to its host node intermittently at which points the required information is exchanged. These points will be referred to as synchronisation points. This section describes the software components of the host that facilitate such synchronisation securely. The next section will consider the corresponding components of the mobile node.

The host node has a TCC similar to the TCCs of other sites. However, this TCC is extended to support the mobile node. This extended TCC will be referred to as an HTCC *(host TCC)*. We will argue in section 5 that the existence of an HTCC in the federation will have a negligible impact on federation members who are not interested in mobile computing. On the other hand, federation members who are willing to allow information onto a mobile node will have to make provision for this. Such provisions can easily be made—see section 5.

The first function of the HTCC is to ensure that the user of the mobile computer is entitled to access any portion of an object that is transferred to the mobile computer. This implies that the HTCC is required to attempt to modify any facets of an object that cannot be transferred in its original form to the mobile computer; such facets have to be modified to a form that has equivalent functionality, but that can be transferred. The HTCC also acts as a repository for those facets of an object that are not relocated to the mobile computer. When objects are reintegrated from the mobile computer with information on the fixed database, the HTCC ensures that changes made to the object during disconnected were authorised.

The assumption has been made that objects to be used on the mobile computer will be relocated to the mobile computer before they are to be used. This was done to increase availability since the mobile computer is only connected to its host intermittently. It is a function of the HTCC to obtain permission from the home site of an object before it is transferred to the mobile computer. Note that, if the mobile computer is connected to its host when a request (user message)

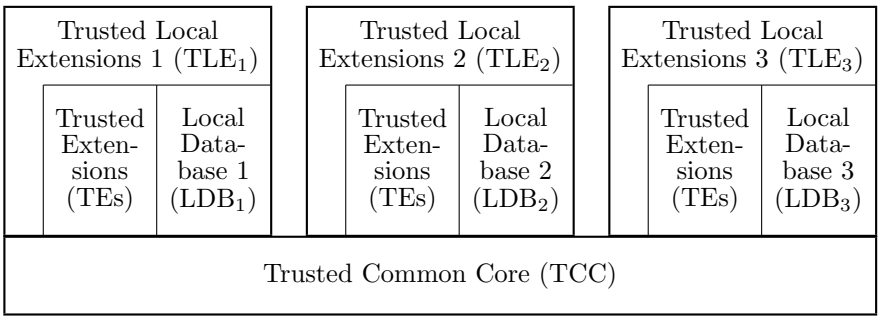

Figure 1. The components of SPO

is sent to an object not located on the mobile computer, the HTCC can relay it to the object like any other TCC. Obviously, this is a solution when access to a sensitive facet is required. If the mobile computer can connect to its host whenever required, this solution ensures access. However, in other cases such messages may have to be queued until a connection is established. The final function of the HTCC is to accept such messages that cannot be handled immediately and process them at the next synchronisation point.

Before we illustrate each of these HTCC functions with an example, it is necessary to state what we mean by *object modification*: An object can be modified to an equivalent object (from the viewpoint of the mobile user) that does not contain information the particular user is not authorised to access. Such a modified object can be transferred to a mobile computer with less risk, if the mobile system is not sophisticated enough to shield portions of the object from the user.

An object consists of a number of variables and methods. To modify an object O for use by a user X a new object O' is created for this purpose. O' is constructed such that

- If v_i is a variable of O that X is authorised to access, O' contains a similar variable v_i' with an initial value equal to the current value of v_i.
- Similarly, if m_i is a method of O that X is authorised to access then O' contains a similar method m_i'.

However, if any of the following conditions hold for a method m_i no corresponding method m_i' will be created for O' even though X may be authorised to access m_i in O:

- If m_i uses any method m_j where m_j' is not in O';
- If m_i accesses a variable v_k where v_k' is not in O'; or
- If m_i performs a calculation where X is not supposed to determine the formula used by m_i (*ie* if m_i contains sensitive procedural knowledge).

If a variable v_i' of O' is not used by any method m_j' of O', v_i' is removed from O'.

Enhancements to object modification will be considered in section 5.

We will now consider the example of the sales representative earlier to describe the functions of the HTCC in more detail. Assume that the sales representative is about to visit a client and needs access to product information—amongst others information about widgets contained in the object WIDGET from the Acme Company (at site ACME). The first function of the HTCC is to transfer those facets of the objects required by the representative to the mobile computer. The normal functionality of TCCs enables them to relocate objects by 'packing' the objects in a transfer format and then transmitting them to the new site. ACME will therefore pack WIDGET and transfer it to the HTCC. In the simplest case the HTCC will transfer the object, as received from ACME, to the mobile computer, where it will be unpacked.

If the entire WIDGET cannot be transferred to the mobile computer because it contains portions the representative is not authorised to access, the HTCC has to modify the object according to the algorithm given above. This means that the form in which the HTCC receives objects has to contain enough information for that algorithm to be applied. The format currently used in the SPO prototype [15] does contain the required information. To illustrate, assume that WIDGET contains GETDESCRIPTION and GETMANUFACTURINGPROCESS methods. In this case the representative is likely to have access to the first method, but not to the second. The second method (and the other related facets) will therefore be removed from WIDGET (and retained at the HTCC) before the modified WIDGET is transferred to the mobile computer. The HTCC has to request access on behalf of the mobile user for each facet of the object to determine if the user is authorised to access that facet. Those facets for which access is denied will be removed according to the process described above.

Note that, if the HTCC requests access on behalf of the mobile user to cache an entity and the entity is protected by time or location depended restrictions the worst case has to be assumed: If a user is, for example, only authorised to access the entity during office hours, it does not mean that the entity can be cached during office hours because it is then potentially accessible outside office hours.

Rather than modifying an object itself, it is preferable that the HTCC requests a suitable object from

the object's home site. In the case of WIDGET it would therefore be better if ACME maintains a version suitable for use by sales representatives and another suitable for use by the engineers and marketing staff of Acme. This allows more control by Acme over exactly what can be relocated to a mobile computer and what not. This is considered in more detail in section 5.

After using WIDGET on the mobile computer, it can be relocated back to the (fixed) federated database. Assume WIDGET has GETDETAILS, SETDETAILS and PLACEORDER methods. It is likely that the SETDETAILS method will not be accessible by the mobile user and therefore would not have been transferred to the mobile computer. However, an unscrupulous mobile user may gain access to the variables containing the details and modify them directly. (Note that these variables have to be present since the mobile user presumably has access to them via GETDETAILS.) Such unauthorised modifications may be possible because the operating system on the mobile computer may not provide adequate protection, or even because the hard disk may be removed, modified on another (unprotected) machine and then replaced in the original computer. In order to prevent such unauthorised modification the methods sent by the mobile user can be logged and replayed on a copy of the concerned object(s) retained at the host. Access controls can again be performed during replay, ensuring that messages are indeed the only mechanism used to modify objects and that only authorised messages are processed—such as PLACEORDER in this example.

The same logging approach can be used to handle messages to objects that have not been transferred to the mobile computer (possibly because they are too sensitive)—such messages are queued until the next synchronisation point. Since modifications to the objects transferred to the mobile computer are effected by replaying the concerned messages this can be done by simply inserting queued messages at the appropriate points in the replay stream. The semantics of when queued messages are logically executed need further attention and will be addressed in section 4.

The structure of an SPO federation with a mobile node is depicted in figure 2. The figure depicts two fixed nodes and one mobile node. The HTCC has been described in this section. The MTC *(mobile trusted core*) is the counterpart of the HTCC on the mobile computer and will be discussed in the next section. The MLDB denotes the local database on the mobile computer.

4 The mobile trusted core

The security related functions of the mobile computer are located in a module referred to as the *mobile trusted core* (MTC). To some extent the MTC functions are mirror images of the HTCC functions and can therefore be discussed briefly. Remember that we assume that the MTC is only partly trusted—see the introduction.

The first function of the MTC is to receive objects sent to it from the HTCC. It is not necessary to describe this function in more detail. Secondly, the MTC is responsible for handling user requests (often disconnected). Such requests will usually involve only objects available on the mobile computer, but sometimes also involve objects not available. Thirdly, objects have to be transferred back to the HTCC at synchronisation points. Lastly, despite the assumption that not much trust can be placed in the MTC, some steps may be taken to increase the level of trust placed in it. These issues are discussed in more detail next.

Handling user requests

A user request here is a message sent by the mobile user to an object. Such messages are handled by the methods of the targeted objects. If those objects and methods are available on the mobile computer, such messages are handled immediately. If disconnected it will be preferable for the mobile computer to only accept messages that can be completed disconnected, but messages to other objects can be accepted and queued. In any case any message sent by the user is logged.

If messages to non-available objects are allowed it is necessary to decide on when such messages are handled logically. To illustrate, assume a sales representative sends a message to change a client's details. Assume further that some details are not cached at the mobile computer. In particular, assume that the method that updates the details first updates the details that are kept on the mobile computer (say NAME), then details that are not kept on the mobile computer (say INCOME) and then again details that are available on the mobile computer (say TELNO). To ensure integrity only the following options exist:

1. Do not allow sending of the CHANGEDETAILS message since not all concerned facets are available.
2. Logically handle the message at the point where it is sent. This means that the user will expect the changes made to influence subsequent (related) operations. Here the CHANGEDETAILS method will have to be suspended when the inaccessible details (INCOME) are to be accessed. A subsequent operation that requires the use of TELNO will have to be queued since previous changes to it are not reflected yet. It may be possible to rearrange the actions of a method so that it first updates all the available variables and then those only at the HTCC, but this is not possible in general. To implement this option it is necessary to lock objects (or instance variables) that have not yet been accessed by a suspended method. However, infor-

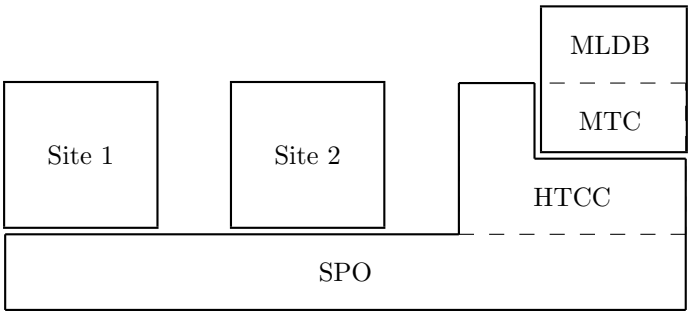

Figure 2. A mobile node in the federation

mation about which objects (or variables) are still to be accessed are not available automatically and special provision for this has to be made—see section 5.

3. Logically handle messages that need to access non-available information at the next synchronisation point. If designed with this in mind, the system will recognise such messages immediately and such a message can be queued and the user notified. Alternatively, messages can be handled and, if they attempt to access non-available information, rolled back, queued and the user notified that the message will be handled later.

An implementation strategy may use any one or more of the options above.

Mobile security

Although the assumption has been made that the mobile computer cannot be trusted, a number of measures can be taken to prevent (or lessen the likelihood of) some attacks.

Firstly, if the operating system on the mobile computer does not provide access control an access control package may be installed. Although such packages on small computers are often easily bypassed, it may prevent the casual passer-by accessing the mobile computer.

Equipping an otherwise untrusted mobile computer with a smartcard or similar intelligent token may also prove to be beneficial. If the log of messages sent on the mobile computer is kept on the smartcard (and writing the log requires a password) the HTCC can be assured that all messages in the log were sent by the mobile user. Similarly, if the values of instance variables are kept on the smartcard and require a password to be accessed, confidentiality and integrity can be ensured. However, space available on a typical smartcard as well as typical communication speed do not always make this a practical solution. Space problems can to some extent be addressed by using the smartcard as an encryption device and/or to store authentication codes (checksums) on the smartcard. However, the solutions mentioned do not address protection of inherently sensitive methods; neither do they prevent Trojan Horse attacks.

Obviously encrypting information kept on the hard disk of the mobile computer protects it from a number of possible attacks if the mobile computer is stolen—this applies whether a smartcard is used or not.

5 Discussion

This section discusses the impact that the proposed mechanisms have on the other nodes in the federation. It is necessary to distinguish between those nodes that want to accommodate mobile users and those that do not.

Federation sites that do not want to accommodate mobile users simply have to refuse relocation of objects to mobile nodes. One function of the TCC is to give permission before objects can be relocated to any other site. A site that does not wish to have objects relocated to a mobile node can simply maintain a list of mobile nodes and refuse relocation to those nodes. Alternatively (and more probably) such a site can maintain a list of sites which it is willing to have its objects relocated to (obviously excluding mobile nodes) and only give permission for its objects to be relocated to those sites. A site that prevents its objects to be relocated to such a site does not prevent the objects from being used—in the case of a mobile user the mobile computer will have to be connected at the point the object is accessed.

Sites that are willing to accommodate mobile computers can do this in a number of ways. In addition to allowing its objects to be relocated to a mobile node, a site can maintain various versions of sensitive objects for different users so that a version exists that can safely be entrusted to each potential (authorised) mobile user. Profiles of tasks performed by each user can also be maintained to simplify the task of deciding exactly which objects (and even facets of objects) have to be relocated to a mobile node at any point. These possibilities will now be discussed briefly.

Enhanced object modification

Ideally each site will be able to modify objects before sending them to an HTCC for relocation to a mobile computer.

This makes it possible for a site to support alternative implementations of objects when required. To illustrate, an item in the catalog of a supplier may normally be implemented as an object with the cost price as an instance variable and a method SELLPRICE that adds, say, 25% to the cost price. Assume that a sales representative is not supposed to know how the selling price for this item is calculated, but is authorised to use the SELLPRICE method. If this method is cached at an untrusted machine the sales representative may gain access to the implementation of the method and determine the formula for selling price calculations. However, the site may support an alternative implementation for this object that stores the selling price in an instance variable and where SELLPRICE merely obtains the value from the instance variable. Normally this implementation would not be used, but when the object is to be transferred to a mobile computer, a copy of this implementation can be instantiated, the selling price loaded from the value obtained by calling the original SELLPRICE and then locking the original implementation until the HTCC and mobile computer are synchronised. Since cost prices are relatively static, this solution will usually be adequate. (Compare this solution to snapshots [2, pp489–490].) If required, the SELLPRICE method of the new object can include an expiry date after which it will stop to function or warn the user that the supplied information cannot be relied on.

Note that the algorithm given in section 3 can also be used to automatically construct suitable versions of most objects by the individual sites.

When objects are analysed for object modification purposes, a list of other services their facets rely on can also be generated. Such a 'requirements' list can be used by the HTCC to transfer other required objects to the mobile computer so that a complete service is available on the mobile computer. If not all such required objects can be transferred to the mobile computer, this list can be used by the MTC to determine whether a message sent on the mobile computer while it is disconnected will be able to complete execution in the disconnected state, before initiating the message.

Task profiles

A task profile will specify exactly which objects are required by a given user to perform a given task on a mobile computer. This information is necessary to decide which objects have to be transferred to a mobile computer for a user to be able to complete the task even if the mobile computer is disconnected—in other words, to increase availability.

Compiling a task profile also involves some analysis: A task can be described by a list of messages that the user may have to send. However, each message that can be sent will cause a method to be activated that can then send a number of messages. It is therefore necessary to determine the transitive closure of all messages needed to complete a task to determine the methods and instance variables required by the user.

The possibility of using workflow scripts to simplify the compilation of task profiles is currently being investigated, and will be reported on in future work. The use of agreements [1] also has potential to be of use here.

6 Conclusion

This paper described an approach to achieve security in a mobile computer that forms part of a federated database. The implementation strategy balanced the confidentiality, integrity and availability requirements of mobile applications. Availability (from the viewpoint of the mobile user) requires that as much as possible information is transferred to the mobile computer. Confidentiality, on the other hand, requires that as little as possible sensitive information is transferred to the mobile computer. This conflict has primarily been addressed by taking the authorisation and requirements of the specific user of the mobile computer into account when transferring an object to the mobile computer. The possibility to modify an object before transferring it to the mobile computer was also considered. Integrity requires that information can only be modified in appropriate ways. The described approach limits effects of inappropriate modifications by the mobile user to that user alone. Approaches to protect the mobile computer from unauthorised use (and, in particular, from unauthorised information modification) were also considered.

Future work includes consideration of the effects of more frequent synchronisation between the mobile computer and the HTCC (made possible by cellular technology), a more detailed study of the potential of intelligent tokens and a study of automated and improved object modification.

References

1. B T Blaustein, C D McCollum, A Rosenthal, K P Smith and L Notargiacomo, "Autonomy and Confidentiality: Secure Federated Data Management", 59–68 in A Motro and M Tennenholtz (eds), *Proceedings of the Second International Workshop on Next Generation Information Technologies and Systems*, Naharia, Israel, June 1995
2. C J Date, *An Introduction to Database Systems, Sixth Edition*, Addison-Wesley, 1995
3. S B Davidson, H Garcia-Molina and D Skeen, "Consistency in Partitioned Networks", *ACM Computing Surveys*, **17**(3):341–370, 1985
4. M H Dunham and A Helal, "Mobile Computing and Databases: Anything New?", *SIGMOD*

Record, **24**(4):5–9, 1995
5. A Elmagarmid, J Jing and T Furukawa, "Wireless Client/Server Computing for Personal Information Services and Applications", *SIGMOD Record*, **24**(4):16–21, 1995
6. D M Huizinga and K A Heflinger, "Two-level Client Caching and Disconnected Operation of Notebook Computers in Distributed Systems", *ACM SIGICE Bulletin*, **21**(1):9–14, 1995
7. T Imielinski and B R Badrinath, "Data Management for Mobile Computing", *SIGMOD Record*, **22**(1):34–39, 1993
8. T Imielinski and B R Badrinath, "Mobile Wireless Computing: Challenges in Data Management", *Communications of the ACM*, **37**(10):18–28, 1994
9. W Kim (ed), *Modern Database Systems: The Object Model, Interoperability and Beyond*, Addison-Wesley, 1995
10. T F Lunt, "Authorization in Object-Oriented Databases", 130–145 in [9], 1995
11. M S Olivier and S H von Solms, "A Taxonomy for Secure Object-oriented Databases", *ACM Transactions on Database Systems*, **19**(1):3–46, 1994
12. M S Olivier, "Self-protecting Objects in a Secure Federated Database", in D L Spooner, S A Demurjian and J E Dobson (eds), *Database Security IX: Status and Prospects*, 27–42, Chapman & Hall, 1996
13. M S Olivier, "Supporting Site Security Policies for Members of Federated Databases", *Fourth European Conference on Information Systems*, Lisbon, July 1996
14. M S Olivier, "Using Workflow to Enhance Security in Federated Databases", in P Horster (ed), *Communications and Multimedia Security II*, 60–71, Chapman & Hall, 1996
15. M S Olivier, "Self-Protecting Objects: A Prototype", *In preparation*, 1997
16. M T Özsu and P Valduriez, *Principles of Distributed Database Systems*, Prentice-Hall, 1991
17. F Rabitti, E Bertino, W Kim and D Woelk, "A Model of Authorization for Next-Generation Database Systems", *ACM Transactions on Database Systems*, **16**(1):88–131, 1991
18. M Satyanarayanan, "Distributed File Systems", 353–383 in S J Mullender (ed), *Distributed Systems, Second Edition*, Addison-Wesley, 1993
19. A P Sheth and J A Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, **22**(3):183–236, 1990
20. B Thuraisingham, "Security issues for federated database systems", *Computers & Security*, **13**:509–25, 1994

M. S. Olivier, "Secure mobile nodes in federated databases," *South African Computer Journal*, **20**, 11–17, 1997.

Preprint
Source: http://mo.co.za