<table>
<thead>
<tr>
<th>South African Computer Journal</th>
<th>Suid-Afrikaanse Rekenaar-tydskrif</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number 20</td>
<td>Nommer 20</td>
</tr>
<tr>
<td>December 1997</td>
<td>Desember 1997</td>
</tr>
</tbody>
</table>

**Computer Science and Information Systems**

**Rekenaarwetenskap en Inligtingstelsels**
For two reasons, this edition of SACJ is far later than it ought to have been. The first reason is that there have been some personnel changes in the editorial team. Lucas Introna, having continued for some time as IS editor after transferring to London, asked to be relieved of his duties. Niek du Plooy has kindly agreed to fulfill this role in a temporary capacity until a suitable replacement for Lucas can be found. Due to work pressure, Riel Smit has also withdrawn as production editor, and has been replaced by John Botha. SACJ owes the two retired members a huge debt of gratitude. During his period of tenure, Lucas did sterling work in setting and maintaining a solid standard for IS contributions. Riel put SACJ on a LaTeX path, and has laboured diligently to produce an aesthetically pleasing product. Thanks are also due to Niek and John for their willingness to take over in their respective roles. Until further notice, IS contributors may forward their submissions directly to Niek at his address given on the front inside cover. I shall put successful authors in touch with John for further instructions regarding final preparation of their manuscripts.

The second reason for a delay in this edition has to do with authors who have not scrupulously followed guidelines for producing their final submissions. There have been a variety of problems ranging from missing citations and inappropriate production of figures to incompatible electronic file submissions. All of this, coupled with our new production editor (who—despite an extremely busy schedule—has valiantly climbed a steep LaTeX learning curve) has resulted in an edition that should have been out to press several weeks earlier.

The editorial team will be giving attention to the general matter of format and submission procedures in future. SACJ's citation and reference methods are somewhat archaic and will probably be revised. All the necessary information will be provided on the new SACJ web site at www.cs.up.ac.za/sacj/. The site will also contain abstracts of articles in this and future editions.

These are times of conflicting stresses on both the academic and industrial IT communities. They are being felt somewhat more acutely in Southern Africa (and presumably in other developing countries) than in the developed world. Internationally there is a tendency to cut back on state financing of universities and a seemingly insatiable demand for IT graduates. Many companies snap up new graduates at attractive salaries, positively discouraging full-time postgraduate studies. International recruitment agencies scour the South African scene for qualified candidates, luring some of our most promising young professionals out of the country. Job-hopping, a drift from academia to industry and from local industry to USA or European industry seems to be the order of the day. Despite the availability of private colleges and institutes, virtual or otherwise, there is a rush of students to university and technikon IT departments, all hoping to get at the IT honey-pot. University administrations are struggling to correct the structural deficiencies of the past and to provide IT departments with sufficient resources to cope with demand. As editor of SACJ, I have no particular competence authoritatively to sum up or analyze these tendencies, but it does seem to me desirable that someone ought to do so. Bodies such as SAICSIT, the CSSA, university authorities, IT industry and state representatives ought actively to pursue joint strategies to ensure that our IT departments are properly resourced and that (non-Zuma) measures are taken to retain graduates in the country. It seems almost redundant to attempt to spell out the consequences of inactivity.

Derrick Kourie
EDITOR
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 access control 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
inherrently 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 var­
ious 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 informa­
tion owned by any number of organisations sharing the fed­
erated database should be accessible from, or relocat­
able 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 fed­
erated database and is trusted by all members of the feder­
ation. 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 infor­
mation about users, user groups and other information re­
quired 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 multi­
policy federated databases, based on the notion of agree­
ments between sites.

3 The host node

The scenario that has been sketched in the previous sec­
tions used a federated database with a mobile node associ­
ated 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 mo­
bile 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 intermit­
tently at which points the required information is ex­
changed. These points will be referred to as synchroni­
ation points. This section describes the software compo­
nents of the host that facilitate such synchronisation se­
curely. 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 por­
tion of an object that is transferred to the mobile computer.
This implies that the HTCC is required to attempt to mod­
ify 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 operation 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 in­
crease availability since the mobile computer is only con­
nected 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 re­
quest (user message) 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 can­
ot 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 trans­
ferred 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 meth­
ods. 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 \( O' \);
mobile user to cache an entity and the entity is protected by
rised to access that facet. Those facets for which access is
time or location depended restrictions the worst case has
from WIDGET (and retained at the HTCC) before the mod­
for each facet of the object to determine if the user is autho­
above.

- 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 consid­
ered in section 5.

We will now consider the example of the sales repre­
sentative 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 mo­
bile computer because it contains portions the representa­
tive 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 proto­
type [15] does contain the required information. To
illustrate, assume that WIDGET contains GETDETAILS
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 mod­
ified 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 autho­
rised 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 PLACE­
ORDER 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 di­
rectly. (Note that these variables have to be present since
the mobile user presumably has access to them via GET­
DETAILS.) 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 (unpro­
tected) machine and then replaced in the original com­
puter. 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 mecha­
nism used to modify objects and that only authorised mes­
ages are processed—such as PLACEORDER in this exa­
ample.

The same logging approach can be used to handle mes­
ages to objects that have not been transferred to the mo­
bile 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 mes­
ages 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

<table>
<thead>
<tr>
<th>Trusted Local Extensions 1 (TLE1)</th>
<th>Trusted Local Extensions 2 (TLE2)</th>
<th>Trusted Local Extensions 3 (TLE3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trusted Extensions (TEs)</td>
<td>Trusted Extensions (TEs)</td>
<td>Trusted Extensions (TEs)</td>
</tr>
<tr>
<td>Local Database 1 (LDB1)</td>
<td>Local Database 2 (LDB2)</td>
<td>Local Database 3 (LDB3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trusted Common Core (TCC)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. The components of SPO

\[ \text{Figure 1. The components of SPO} \]
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, information 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

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use wide margins and 1\$\frac{1}{2}\$ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author’s initials and surname;
  - author’s affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text in square brackets [1-3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As a \LaTeX\ file(s), either on a diskette, or via email/ftp—a \LaTeX\ style file is available from the production editor;
2. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be original line drawings/printouts, (not photocopies) on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
   - Contact the production editor for markup instructions.
3. In exceptional cases camera-ready format may be accepted—a detailed page specification is available from the production editor;

Authors of accepted papers will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30-00 per final page for \LaTeX\ or camera-ready contributions that require no further attention. The maximum is R120-00 per page (charges include VAT).

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in category 2 above may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author’s responsibility to ensure that such submissions are error-free. Camera-ready submissions will only be accepted if they are in strict accordance with the detailed guidelines.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

### Contents

**Editorial**

DG Kourie ......................................................... 1

**Research Contributions**

The Abstraction-First Approach to Encouraging Reuse  
P Machanick ..................................................... 2

Secure Mobile Nodes in Federated Databases  
MS Olivier .......................................................... 11

Word Prediction Strategies in Program Editing Environments  
I Sanders and C Tsai ........................................... 18

A Computerised-consultation Service for the Computerisation of the Very Small Small-business Enterprise  
CW Rensleigh and MS Olivier .................................. 25

Some Typical Phases of a Business Transformation Project: The First Steps Toward a Methodology?  
D Remenyi ............................................................ 36

**Technical Reports**

Theory Meets Practice: Using Smith’s Normalization in Complex Systems  
AJ van der Merwe and WA Labuschagne ....................... 44

Applying Software Engineering Methods to Instructional Systems Development  
P Kotzé and R de Villiers ........................................ 49

**Communications and Viewpoints**

Mobile Agents at ISADS 97  
I Vosloo ............................................................. A57

The Recovery Problem in Multidatabase Systems—Characteristics and Solutions  
K Renaud and Paula Kotzé ........................................ A62