11

Alter-egos and Roles — Supporting Workflow Security in Cyberspace

E. Gudes
Ben-Gurion University
Beer-Sheva, Israel
ehud@bengus.bgu.ac.il

R.P. van de Riet
Vrije Universiteit
Amsterdam, Holland
vdriet@cs.vu.nl

J.F.M. Burg
Vrije Universiteit
Amsterdam, Holland
jfmburg@cs.vu.nl

M.S. Olivier
Rand Afrikaans University
Johannesburg, South Africa
molivier@rkw.rau.ac.za

Abstract

Workflow Management (WFM) Systems automate traditional processes where information flows between individuals. WFM systems have two major implications for security. Firstly, since the description of a workflow process explicitly states *when* which function is to be performed by *whom*, security specifications may be automatically derived from such descriptions. Secondly, the derived security specifications have to be enforced. This paper considers these issues for a Cyberspace workflow system by describing a small, but comprehensive example.

The notion of an Alter-ego is central in this description: Alter-egos are objects that represent individuals in Cyberspace (and not merely identify them). In Cyberspace, documents in a workflow system therefore flow between Alter-egos, rather than between individuals.

Keywords

Security and Database systems, Workflow, Cyberspace, Object-Oriented Databases, Role-based security

1 INTRODUCTION

Workflow Management (WFM) Systems automate traditional processes where information flows between individuals. Although WFM systems have been in existence for a number of years, the trend towards greater interconnection will greatly impact such systems. On the one hand, interaction will involve more and more nonhuman participants. On the other hand the participants

Database Security XI T.Y. Lin and Shelly Qian (Eds)

in workflow processes will become more and more unrelated. To illustrate the latter trend, consider the following 'generations' of workflow systems:

1. All individuals who take part in a workflow process are typically part of the same organization.
2. Individuals who take part in a workflow system are not necessarily members of the organization who 'owns' the WFM system, but are registered with that organization.
3. Participants in a workflow process may never have had any contact prior to participating in the same workflow process.

The key to secure implementation of later generation WFM systems is proper authentication and authorization of participants in a workflow process. It is our contention that Alter-egos (see the next section) are particularly suitable for authentication, while roles are particularly suitable for authorization. Stated differently: we will assume that a potential participant will present an Alter-ego that will serve as proof of the participant's identity.

The intention of this paper is to study the derivation of security rules from a WFM design tool and to consider how such rules may be implemented. Of particular concern is the distinction between the individual, represented by an Alter-ego, and the role in which the individual acts.

The paper is structured as follows: The following section gives some background on Alter-egos, workflow and security. Section 3 introduces the insurance claim example that is used in this paper and shows how security specifications may be derived from the workflow specification. Section 4 discusses an implementation strategy for a workflow process. Section 6 contains the conclusions of the paper.

2 BACKGROUND

2.1 Alter-egos

Individuals, either in an office environment, or in their homes, will be represented in Cyberspace by objects, called Alter-egos, in the sense of Object-Oriented Technology, and may be considered a combination of Social Security Number and e-mail address. They were introduced in (van de Riet & Gudes 1996), where it was shown how these Alter-egos can be structured and how Security and Privacy (S&P) aspects can be dealt with. Questions around Responsibility and Obligations of Alter-egos have been discussed in (van de Riet & Burg 1996a, van de Riet & Burg 1996b).

The use of Alter-egos to provide high-level security has been discussed in an earlier paper (van de Riet & Gudes 1996). The main idea was that if the underlying communication system of Cyberspace ensures that every message

contains an unforgeable Alter-ego of the sender (or initiator), one can design more powerful and higher level protection mechanisms than those existing today and which rely mainly on encrypting messages.

2.2 Workflow

In Workflow management (WFM) applications there are tasks to be completed by some organization, but the organization procedures require that this task will be carried out in steps where each step is executed by a different individual and no step can be performed before the steps it depends on are completed (Georgakopoulos *et al.* 1995). We shall demonstrate a certain WFM-tool, COLOR-X, developed by the group in Amsterdam to model Information and Communication Systems, using linguistic knowledge, and we will see how S&P rules can be derived from COLOR-X diagrams.

WFM tools are currently being used to specify how people and information systems are cooperating within one organization. There are at least three reasons why WFM techniques are also useful in Cyberspace. First, organizations tend to become multi-national and communication takes place in a global manner. Secondly, more and more commerce is being done electronically. This implies that procedures have to be designed to specify the behaviour of the participants. These procedures may be somewhat different from ordinary WFM designs, where the emphasis is on carrying out certain tasks by the users, while in commerce procedures are based on negotiating, promises, commitments and deliveries of goods and money. However, as we will see, these notions are also present in the WFM tool we will use. Thirdly, people will be participants in all kinds of formalized procedures, such as tax paying or home banking.

2.3 Workflow and Security

This being said, how can we derive security and privacy rules from the Workflow diagrams (WFDs)? Specifying tasks and actions of people working in an organization naturally also involves the specification of their responsibilities (van de Riet & Burg 1996a, van de Riet & Burg 1996b, Olivier 1996). This is what WFDs usually do. Responsibility implies access to databases to perform certain actions on data of individuals.

A Workflow Authorization Model is proposed in (Atluri & Huang 1996). Authorization Templates are associated with each workflow task and used to grant rights to subjects only when they require the rights to perform tasks. A Petri net implementation model is also given. Where (Atluri & Huang 1996) focusses on synchronising workflow and authorization flow, the current paper focusses on the relationship between individuals (represented by Alter-egos)

and the roles they occupy in such workflow processes, while in (Atluri & Huang 1996) the authors emphasize information-flow issues using a multi-level security model.

2.4 Alter-egos, roles and workflow security

As noted earlier, it is our contention that Alter-egos are particularly suitable for authentication, while roles are particularly suitable for authorization. Not only are Alter-egos suitable for authentication — the nature of the mechanism to represent participants *has* to progress towards full-function Alter-egos. Similarly, the role concept needs to evolve from that required by the earlier generations to that required by the later generations.

Alter-egos required by first generation WFM systems primarily serve to authenticate the user to the system. A user identifier with a password or PIN (personal identification number) may be adequate. In second generation systems the requirements, in addition, include privacy, integrity and non-repudiation. These issues become important since many of the participants are not employed by the organization and are accessing the system from remote systems. In third generation systems these issues are still important, but it also becomes important to construct the Alter-egos such that they can be trusted by the represented individual to act as agent for the individual.

3 THE INSURANCE-CLAIM APPLICATION

In this paper the following simple fragment of a workflow application will be used to illustrate the concepts involved: A *claimant* submits an insurance claim, which is either approved or rejected by some *approver*. If the claim is not approved (or rejected) within some specified period, the approver is reminded to give attention to this claim. The fragment is depicted graphically in figure 1. For another example, see (van de Riet & Burg 1997).

The following requirements are obvious, even from this simple example:

1. The approver has to be duly authorized to approve the claim. The requirements for such authorization depend on the specific application; the mechanisms to enforce such requirements are the concern of this paper.
2. Access to the documents involved in the workflow depend on the roles individuals play in the workflow process.

A specification for the same process, but using a COLOR-X diagram, is given in figure 2. Note that in figure 2 we used the following conventions: A box in figure 2 (a) denotes an entity or type. A line is a relationship and a line

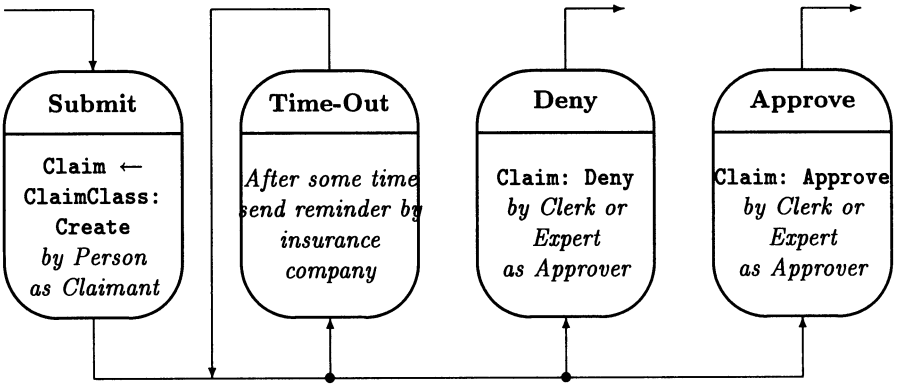

Figure 1 Fragment of a workflow process

with open arrow is an is_a relationship. The circle with an X means exclusion. In figure 2 (b) we have:

- each box of actions has a mode: PERMIT, NEC or MUST. The latter one means an obligation based on some negotiating in the past: as we are not sure that the action is actually carried out within the prescribed time it is necessary to define a counter measure. The mode NEC means we can be sure the action is necessarily carried out by the system. PERMIT is self evident.
- the actions are described in a formal language involving the participants and their roles;
- the lightning arrow denotes a situation in which the conditions in the identification part (denoted by id) are not satisfied. In the diagram we let the reminders go indefinitely, which was done for shortness sake;
- The "approve" and "deny" boxes from figure 1 correspond to the arrows R2="approve" and R2="deny" going out the lowest but one box in figure 2 (b).

We now derive the authorization tuples from the diagrams above. We use the following heuristic rules:

1. If an action involves data in a database, the agent of this action should be authorized to perform the corresponding actions on the database.
2. An action, with modality MUST, involves an obligation to perform a specific action within a prescribed amount of time. This implies that in some database, oblDB, the administration about this obligation is kept. The object which creates the MUST action, i.e. the agent of the action leading to this MUST action, can move the deadline (only shift it to the future of course); that is explicitly not allowed to the object who has to carry out the

action. Of course this object can refuse to carry it out, but then penalties may be the result.

3. In a send or reminder action the sender is assumed to write in the message database, while the receiver is assumed to be able to read from it.

The databases involved are:

- For the claims: claimDB;
- For obligations: oblDB; and
- For the messages: messDB.

The syntax we will use to indicate that an actor in some role is authorized to perform a specific operation on a specific database, is as follows:

```
AUTH <name database, role actor, operation>
```

From the diagram we thus have the following authorization tuples:

```
AUTH<claimDB, claimant, add>
AUTH<claimDB, insurance_company, read>
AUTH<claimDB, approver, read>
AUTH<claimDB, cashier, create>
AUTH<oblDB, claimant, shift>
AUTH<oblDB, approver, not shift>
AUTH<messDB, insurance_company, write>
AUTH<messDB, approver, read>
AUTH<messDB, cashier, read>
```

Furthermore, from figure 2 (b) we derive that there is a partial ordering with respect to authorization for actions concerning the claimDB, for the following roles:

```
clerk >> approver
expert >> approver
```

According to figure 2 (a) all three roles or types are subtypes of employee. Note also that an approver cannot be the same person as the claimant.

The authorization tuples as derived above may in general conflict, as there are positive and negative tuples. In the case of an approver not being allowed to shift an obligation, we also may have the situation that the approver sets a deadline for someone else, in which case there will appear an authorization tuple:

```
AUTH<oblDB, approver, shift>
```

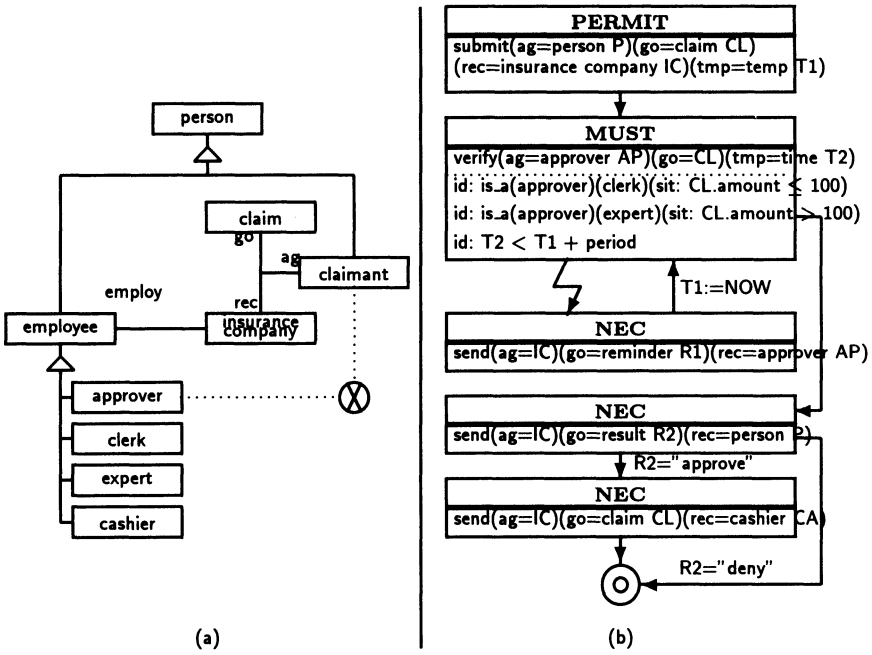

Figure 2 Application specified using a COLOR-X diagram

which is evidently in conflict with the one we mentioned. The reason for this problem is, however, very simple: oversimplification. In actual practice we would also register the specific task to be performed, and not merely allow access to an entire database.

The above security checks represent simple access-control rules which can be automatically derived from the Workflow specifications. However, in reality the situation is much more complex. First, there may be more complex constraints, such as: a claimant and an approver may not be the same individual (alter-ego). The issue of constraints is discussed further in section 4.2. Second, the distinction between a *Role* and *Individual* (Alter-ego) may be important. In some instances, only the role is required to approve access. In other instances, only a specific Alter-ego may approve a claim. This is further discussed in section 4.3. Thirdly, the above rules assume a static state where the agents represented by Alter-egos and Roles are "alive" and respond in time. However, time is an important factor! Rights of some individuals may be time dependent (see also (Bertino *et al.* 1994)). In particular, users may change their role in the organization, new roles may need to be created for a particular alter-ego (e.g. the expert role), or deleted. The administration of roles and its dynamics are therefore an important issue. This is discussed in section 4.4.

4 IMPLEMENTATION

The implementation of the above model largely depends on the underlying object system. Below we sketch our approach to implement it in Mokum. Note the fact that the various agents are responsible for performing the required security checks. This avoids the bottlenecks associated with a centralised checking facility in a distributed system. It also provides more flexibility in the types of checks that may be performed over those from a standardised facility. The fact that the checks are interspersed through the code is not a concern because the intention is to automatically derive the code and the checks from the workflow specification. The following discussion concentrates on implementing the run-time access-control, the administration issues are discussed in Section 4.3.

4.1 Implementation based on Mokum

The main idea here is to use the "Collection Keepers" (van de Riet & Gudes 1996) to execute the customized (knowledge-based) security checks. Some of the Workflow security related pseudo-code performed by each participant (similar to Mokum's code described in that paper) is shown below.

Authorization tuples as described earlier, are defined as follows:

```
type authorization_tuple
    has_a alter_ego: thing
    has_a role: thing
    has_a database: thing
    has_a access_mode: thing
```

The Application Administrator is responsible for performing additional authentication checks and for authorizing operations in the workflow system. In order to do this, the Application Administrator maintains the authorization database and processes the authentication and authorization events. For brevity the details of the code have been omitted.

The insurance company receives the claim and directs it to an approver, which may be a clerk or an expert.

```
type insurance_company is_a thing
  script
    at_trigger submission:
      message:role=claimant,
      A = message:claim:amount,
      (A < 100, choose_clerk(AP); choose_expert(AP)),
      add_type(AP, approver),
```

The claimant is the submitter of the claim.

```
type claimant is_a person
  script
    at_trigger send_claim:
      create(M, messageT, [(data_base=claimDB), (access_mode=add)]),
      send(application_administrator, authenticate_user,M),
      compose (Claim),
      /* using the incident and the amount of money   */
      /* involved by interaction with the user        */
      Claim to M:claim,
      send(application_administrator,authorize_user,M),
      /* Note that the authorization goes before the following */
      /* action, so if it fails the next action will not be    */
      /* executed                                              */
      send (insurance_company,submission, M).
  end_script
```

The approver is the clerk or expert responsible for verifying and approving claims.

```
type approver is_a employee
  script
    at_trigger verify_claim, reminder_message:
      /* note that for the sake of brevity we let the approver */
      /* do the same when it receives a new case or when he    */
      /* receives reminder for some case                       */
      CL=message:claim, C=message:client,
      create(M1, messageT, [(data_base=claimDB),
        (access_mode=read),(claim,message:claim)]),
      send(application_administrator,authorize_user,M1),
      show_claim_on_screen(CL),
      /* now the approver may take a few days to verify the claim*/
      verify_claim(CL, Result),
      create(M2, messageT, [(client=C), (result=Result])]),
      send(insurance_company, receive_from_approver,M2)
  end_script
```

4.2 Constraint checking

Although not shown in the example given in figure 2 (b), it is possible to specify constraints in the WFM diagrams. Such constraints may have the form

"the approver of a claim must be different from the claimant". These constraints can also be specified in the Mokum scripts of the Alter-egos involved.

Constraints may be specified as pre-conditions, post-conditions or through-conditions. Ideally, constraints are checked by the various `choose` methods. For example, when the `choose_expert` method is selected above one could have as a Pre-condition: `not AP = sender`.

In contrast, if the constraint specifies that some particular approver may not approve claims involving a claimant who has not paid any outstanding fees, this may not be verifiable when the approver is selected, since the approval process may take a significant amount of time, in which the claimant's account may become overdue. Such a condition therefore needs to be specified as a post-condition, to be checked at completion of the approval activity. This could be implemented by adding somewhere, after getting the response of the approver and before sending of messages to cashier and claimant, the following to the code of the trigger submission: `check_due_fees(sender)`.

If the constraint specifies that the approver may not be married to the claimant, the intention may be that they should not be married at any point during the approval process (which may take a while!). It is clear that neither a pre-condition, nor a post-condition (or a combination of the two) fully checks this constraint. (A post-condition is, in principle usable, if a history of marriage(s) is kept; it may, however, postpone handling of the situation longer than necessary.) We call these constraints *through-constraints*. In our case they could be implemented in the Mokum script of approver, instead of `verify_claim`, we could have: `verify_claim_and_check_spouse`. Of course in the body of this procedure the actual checking needs to be programmed — for example to inspect a database.

4.3 Roles versus individuals

We assume that messages in a workflow system are routed to any qualifying individual operating in an appropriate role. It is, however, possible that in exceptional cases, the message needs to be directed at a particular individual acting in a role. A particular approver may, for example, be requested to deal with a particular claim because of expert knowledge the approver has. Another example is the "not-married" constraint above.

Where previous constraints dealt with avoiding of conflicts, it is also sometimes necessary that people co-operate to perform a given action. Consider a bank safe that is to be opened only after two distinct, authorised subjects have requested it. It is clear that this requirement is easily handled in the current approach by simply requiring the related actions from two 'keybearers' (who have to be distinct individuals) within an acceptably short time period (enforced with deadlines). The Alter-ego concept gives us this essential capability of distinction between and identifying certain individuals

4.4 Role administration

As mentioned in section 3, a Workflow system is a very dynamic system. Roles need to be created, deleted changed, etc. The administration of roles is therefore a critical component in the implementation. For that purpose we assume the existence of the Workflow security administrator (called above: Application administrator). This administrator handles the following tasks:

1. It authenticates each new user and creates for her the appropriate role.
2. It is consulted whenever the dynamics of the situation requires the change of roles. For example, if for a particular claim, the Expert role is required, the administrator is consulted to check whether the current Approver's alter-ego can amplify its role to become an Expert. If not, it checks whether there is already an active Expert role in the system and if there is, sends to the Approver node its identity, and if it does not exist, it will wait for the right alter-ego to be instantiated and authenticated with this Role.

We see the Roles/Alter-ego administration as an essential part of the implementation. This implementation can use the Distributed Mokum architecture described next.

4.5 Distributed Mokum

A first version of Distributed Mokum (Radu, Dehne & van de Riet 1997) is currently being tested. This version of Mokum can be run at several sites. The objects can communicate with 'local' objects and with 'global' objects. An interface has been written in Java which, using the socket mechanism, cares for the necessary communication. Each Mokum program has its own Administrator, which not only takes care of the external communication problems, but also, beit in a very limited fashion, of security problems. It is a kind of Security Administrator. We are still looking at the problem to concentrate such security administrative duties in an applet. In this implementation an applet is being used for the man-machine interface between a user and the Mokum program.

How to implement a Mokum system which is truly distributed, where objects may consist of subobjects residing at different sites and where addressing these subobjects is completely transparent, is still a subject of study.

5 SUPPORTING TECHNOLOGIES

In order to securely implement the system described above, it is necessary to ensure that the underlying infrastructure is supporting appropriately. Secure

communication protocols are obviously essential. Before we review such protocols, we briefly consider CORBA to support the implementation described above.

5.1 Implementation based on CORBA

CORBA which is becoming one of the major standards for Object-oriented software systems has recently published the Security reference model specifications (OMG 1996). Many of our concepts map directly into the CORBA architecture. Our Alter-ego will represent a "principal" in CORBA. In terms of protecting messages and assuring message integrity, CORBA provides some facilities, but in our opinion these facilities are redundant in Cyberspace because of the existence of secure communication protocols (see section 5.2).

The active customized code for each participant will be coded within the "access decision functions" of CORBA, and is enforced automatically by the ORB (Object Request Broker) before any Object invocation. Auditing can also be specified by the CORBA's model and also implemented by the ORB. In addition Non-Repudiation services are also provided for reasons of Accountability. Although Roles as used in this paper are not supported in the CORBA model, other means such as Domains may be used to implement them. The problem of *administration* of authorization information is discussed in the (OMG 1996), but a full specification is not given.

Since the security reference model is not yet implemented and not even accepted, one can resort to existing CORBA services to implement our model. One such approach can follow the scheme suggested by Sheth in his CORBA based Workflow architecture (Miller *et al.* 1996). In this architecture, there is a Task manager associated with every task (object in our case), which executes the interface code defined in the IDL file. We will need to add to such a task the active security checks and the ability to **re-initialize** its parameters each time it receives a message from the Application administrator requiring it.

5.2 Using Secure communication protocols

The recent literature and the World Wide Web has much information about secure communication protocols. Lipp and Hassler (Lipp & Hassler 1996) present a survey of such protocols. They can be at the message level such as Netscape SSL or Microsoft PCT (PCT 1996) or depend on the requirements of the application which is HTTP or higher, such as SHTTP. SHTTP supports end-to-end secured transactions which can be initiated symmetrically, so that servers and clients are treated equally with respect to their preferences. Message protection may be provided by signing, authenticating, or encrypting the message, or by applying any combination of these. There

are basically two types of messages in the above described Workflow system. There are the standard messages between the participants' Alter-egos. These messages are assumed to contain the Sender's pair *(Role, Alter-ego)* and the receiver's Role. The integrity of messages provided by the lower level protocols is sufficient. Once the Alter-ego is authenticated, and a Role is assigned, the pair *(Role, Alter-ego)* is assumed to be part of every message (e.g. in its header) and this is sufficient for applying the higher-level customized security checks.

The existence of the pair *(Role, Alter-ego)* — actually the triple (Site, Role, Alter-ego) — can also be used by security firewalls. The firewall will be associated with the workflow application and will reject any message which does not have in the header the *(Alter-ego, Role)* pair.

Another problem is the protection of documents which may be required by the Expert system to approve the claim. One need to authenticate the validity of these documents and just message protection is not sufficient. A scheme similar to the one suggested to protect documents on the Web, i.e. CCI-PGP scheme can be used for that (Weeks *et al.* 1996).

6 CONCLUSIONS

This paper considered security in a workflow system. It has been argued that the notion of Alter-egos is central in such workflow systems. The participation of an Alter-ego in each message enables the complete authentication and some specific individual-based checks that are required in such an environment.

It is clear from the previous section that different implementation alternatives do not only hold different advantages, but are based on different trusted components: Note in particular the trust placed in the various actors in the case of the Mokum implementation.

It is clear that the role concept in workflow system holds particular benefits. Additional research needs to be done to investigate this potential — in particular for third generation workflow systems.

REFERENCES

Atluri, V., and Huang, W.K. (1996) An extended petri net model for supporting workflow in a multilevel secure environment, *Proc. Annual IFIP WG 11.3 Conf. on Database Security*, Como, Italy, August, 1996, pp. 199-216.

Atluri, V., and Huang, W.K. (1996) An Authorization Model for Workflows, *Computer Security — ESORICS 1996* (eds. E. Bertino, H. Kurth, G. Martella and E. Montolivo), Springer, 1996, pp. 44–64.

Bertino, E., Bettini, C., and Samarati, P. (1994) A Time-based Authorization Model, *Proc. ACM Int. Conf. on Computer and Communication*

Security, Fairfax, Va, Nov. 1994, pp. 126-135.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995) An overview of workflow management: from process modelling to workflow automation infrastructure, *Distributed and Parallel Databases*, Vol 3, No. 2, 1995, pp. 119-154.

Lipp, P., and Hassler, V. (1996) Security concepts for the WWW, *Proc. 2nd Int. Conf. on Communication and Multi-media security*, Essen, Germany, 1996, pp. 85-95.

MicroSoft Corp. (1996) URL: *http://microsoft.com/intdev/security/misf13_4.htm*

Miller, J.A., Sheth, A.P., Kochut, K.J., and Wang, X. (1996) CORBA-based run-time architecture for Workflow management systems, *Journal of Database Management*, Vol 7, No. 1, Winter, 1996, pp. 16-27.

Olivier, M.S. (1996) Using workflow to enhance security in federated databases, *Proc. 2nd Int. Conf. on Communication and Multimedia Security*, Essen, Germany, 1996, pp. 61-72.

Object Management Group (1993) *The Common Object Request Broker: Architecture and Specification*, OMG Document No. 93.12.1, December, 1991.

Object Management Group (1996) URL: *http://www.omg.org:80/docs/orbos/* Documents: 96-08-03.ps, 96-08-04.ps, 96-08-05.ps, and 96-08-06.ps.

Radu, S., Dehne, F., and Van de Riet, R.P. (1997) *A first step towards distributed Mokum*, Technical Report 428, Computer Science Department, Vrije Universiteit, Amsterdam, In preparation.

Riet, R.P. van de, and Burg, J.F.M. (1996a) Modelling Alter-egos in Cyberspace: who is responsible?, *Proc. of WebNet 96*, San Francisco, 1996, AACE, Charlottesville, USA, pp. 462-467.

Riet, R.P. van de, and Burg, J.F.M. (1996b) Linguistic Tools for Modelling Alter Egos in Cyberspace: Who is Responsible?, *Journal of Universal Computer Science*, Vol 2, No. 9, Springer,1996, pp. 623-636.

Riet, R.P. van de, and Burg, J.F.M. (1997) Modelling Alter-egos in Cyberspace using a Work Flow Management Tool: who takes care of Security and Privacy?, Submitted.

Riet, R.P. van de, and Gudes, E. (1996) An object-oriented database architecture for providing high-level security in Cyberspace, *Proc. 10th Annual IFIP WG 11.3 Conf. on Database Security*, Como, Italy, August, 1996, pp. 92-115.

Weeks, J.A., Cain, A., and Sanderson, B. (1996) CCI-Based Web security: a design using PGP, URL: *http://sdg.ncsa.uiuc.edu/~jweeks/www4/paper/current_rev.html*