A Comprehensive Need-to-Know Access Control System and its Application for Medical Information Systems

R. Holbein,
S. Teufel,
O. Morger,
K. Bauknecht
Department of Computer Science
Winterthurerstr. 190, 8057 Zurich, Switzerland

+41-1-257 43 11

{holbein, teufel, omorger, bauknecht}@ifi.unizh.ch

Abstract

In this paper we present an access control system (ACS) that allows implemention as well as management of comprehensive need-to-know access control policies. The overall system is built around a role based ACS that has been extended by two additional components namely, a security design and a context autentication component which allow the overall system to cohesively implement and manage need-to-know policies. The security design component systematically generates access control information that is appropriate to initialise the role based ACS according to the individual need-to-know within an organisation. The context authentication component on the other hand, has been integrated with the access control decision facility of the role based ACS. It dynamically verifies if a need-to-know really exists at the particular point in time when users request access to information. Finally, we describe an application scenario that illustrates the benefits provided by our need-to-know ACS concerning privacy of patient data within a hospital environment.

Keywords

Access control, business process modelling, business transaction, context authentication, need-to-know, role based access control, security policy, security design, security modelling, medical information systems

1 INTRODUCTION

In this paper we introduce an access control system (ACS) that has been designed in order to prevent information misuse and therefore, allows to protect privacy of sensitive personal information. We also describe an application of this access control system for medical information systems within a hospital. The implementation of our system consists of a number

of components that correspond to the concepts for need-to-know policy implementation presented in earlier papers [Holbein and Teufel 1995] [Holbein et al. 1995] [Holbein et al. 1996] [Teufel and Holbein 1996]. Therefore, our particular implementation and application area provide a synthesis of interrelated research results that realise comprehensive need-to-know access control for privacy of sensitive patient information in a hospital. There are two major aspects that must be considered in order to realise comprehensive need-to-know access controls [Holbein 1996]:

1. a task related specification of access rights that agents within an organisation must receive in order to fulfil their tasks;
2. a task related evaluation of access requests that considers the current business transactions within an organisation.

The task related specification of access rights is necessary to initialise an access control system according to the need-to-know within an organisation's business processes. This part of our overall system is called *security design* component. The task related evaluation of access requests on the other side, is called access control component. This part verifies if a task related access right that supports a particular request was defined during security design and furthermore, considers an organisation's current business transactions to ensure that the users' need-to-know is valid at the point in time when the particular access was requested. The latter is called *context authentication* [Holbein and Teufel 1995]. We have developed security subsystems for access control systems to adequately address these aspects comprehensively. Our subsystems allow a broad range of current access control systems which provide some basic features for need-to-know policy implementation to be enhanced for security design and context authentication. The result of this enhancement is called a comprehensive need-to-know access control system.

In our implementation we integrate the security design and context authentication subsystems with a role based access control system called Argos [Jonscher and Dittrich 1995] and apply the overall system to a mobile hospital bed unit (HBU) [Teufel and Holbein 1996]. The HBU is a hospital bed that includes a device for accessing a medical information system directly from a patient's bed within a hospital. This allows multiple medical, administrative as well as patient services and information to be used from widely distributed and commonly accessible sites. However, this also causes high sensitive patient information to be exposed and therefore, implies threats for privacy that require comprehensive need-to-know access controls.

The remainder of the paper is structured as follows: Chapter 2 continues with a conceptual overview and a description of the overall system architecture. Subsequently, in chapter 3 we focus on the major functionality of the security design and context authentication subsystems in some more detail. In chapter 4 we illustrate the benefits of our system within a hospital environment by means of an information access scenario. Finally, we point out some conclusions and further research activities.

2 CONCEPTUAL OVERVIEW AND SYSTEM ARCHITECTURE

From a generic point of view the conceptual overview of our comprehensive need-to-know access control system consists of two parts that correspond to the two major aspects as introduced above chapter (Figure 1): First, there is a *security design* subsystem that is connected to the access control system for administration of access rights according to a need-to-know policy. The security design system includes a number of interfaces that allow to import business models, e.g. business process models (BPM), to automatically derive security information and to establish need-to-know security models [Holbein et al. 1995] [Holbein et al.

1996]. These need-to-know security models can be exported to a number of access control systems. Again, there are interfaces that translate the security model to specific access control information that can be interpreted by the target system [IBM 1995]. Second, there is a *context authentication* subsystem which can be linked to different access control systems (ACS) on different platforms via corresponding ACS client modules. These modules are hooked up within the access control systems' procedure for evaluation of access requests and allow to verify if a current need-to-know concerning a particular access request exists [Holbein and Teufel 1995]. For that purpose it includes multiple server interfaces for different business automation systems with the capability to provide information concerning the state of an organisation's current business transactions, for example, workflow management systems[1].

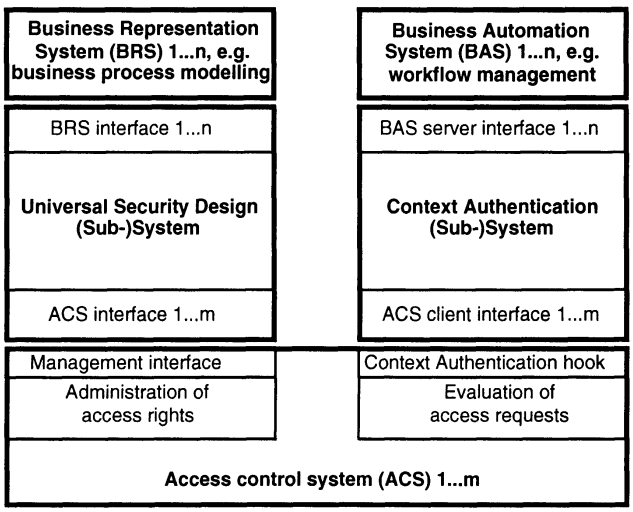

Figure 1: Conceptual overview

Our system implementation is based on the conceptual foundation above. It is constituted by the components shown in Figure 2. Below, there is a summarised description of all these components. The FELIX security design and CARDS context authentication components will be described in more detail in sections 3.1 and 3.2. The other components which are not for further consideration will be described as far as necessary in order to understand the overall architecture according to Figure 2:

- *ActionWorkflow Builder*: Business process modelling tool from Action Technologies which provides *transaction based BPM* as input for the FELIX security design system. The ActionWorkflow system is a software package from ActionTechnologies that provides tools for business process modelling and workflow control. One component within this tool set is called ActionWorkflow Application Builder (AW Builder). It allows transaction based BPM to be specified on a high level of granularity and detail. This tool runs on a Windows95-PC and communicates via a TCP/IP network connection with the BPM-definition database that runs on a OS/2-PC.

---

[1] In general, it is very useful when BRS and BAS are interrelated for example, workflow management systems that usually include a business process modelling component as well as a workflow control engine. However, this is not necessary in principle.

- *ActionWorkflow Manager*: Workflow management system from Action Technologies that controls an organisation's business transactions according to BPM definitions provided by the ActionWorkflow Builder. This system runs on a OS/2-PC with a proprietary database that can be accessed via a Lotus Notes interface.

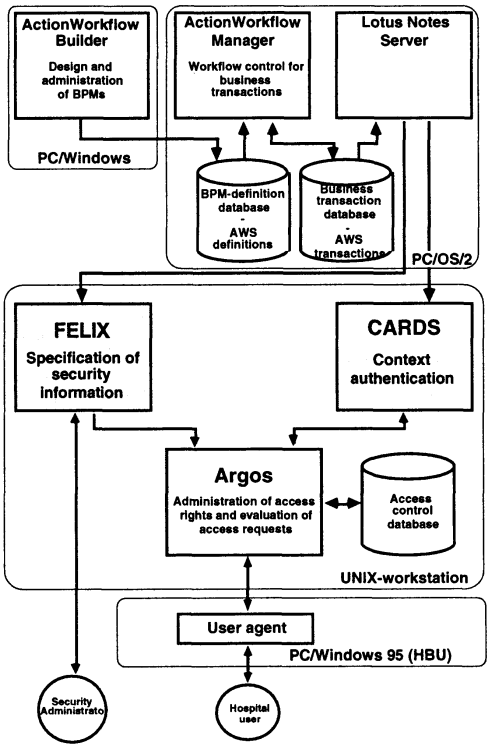

Figure 2: Overall system architecture

- *FELIX security design system*: Security design environment providing automation of our need-to-know security design method. This will be described in section 3.1.
- *CARDS*: Context Authentication service for Role based access control in Distributed Systems. This will be described in section 3.2.
- *Argos access control system*: Discretionary access control system providing role based mechanisms. This powerful ACS is highly flexible and configurable and was developed by Jonscher and Dittrich as a global access control system for heterogeneous database federations [Jonscher and Dittrich 1995]. Argos is founded on an object oriented data model [Jonscher and Dittrich 1993] and implemented on a UNIX platform as C++ application using an ObjectStore database[2]. Extensive customisation according to a security policy is possible due to a number of parameters that configure the reference monitor, e.g. closed/open world assumption. There are also capabilities to establish global controls, i.e. mandatory extensions according to an administration paradigm. The basic characteristics of

2 Argos was implemented as a prototype within the SPP-IF project CHASSIS [Jonscher and Dittrich 1995]. CHASSIS = Configurable Heterogeneous And Safe, Secure Information Systems; SPP-IF No. 5003-34355.

Argos and the mechanisms which FELIX uses to prepare for an implementation of individual need-to-know security models are summarised below.

Basically, Argos is characterised by discretionary specified rights which can be held by authorisation units. These authorisation units are represented by subject expressions. A *role* concept allows abstract subject expressions to be specified which represent users or sets of users according to their position - maybe a functional, organisational or social one - within a domain. Roles can be related to each other with a subordination relationship where permissions are inherited from subordinated roles and prohibitions from superior roles. Individual users can be associated with roles. Combinations of users and/or roles are also allowed to represent authorisation units. These combinations are called complex subjects. Consequently, access rights can be granted to subjects which may be individual users, roles or combinations of the previous ones, i.e. complex subjects. Furthermore, association as well as activation conflict relations can be used to establish restrictions for user-role association or simultaneous activation of roles during system operation.

A concept of *domains* is available for subjects as well as protection objects. Domains allow grouping and even nesting of sets of subjects and protection objects. Therefore, the definition of access rights refers to the whole set of elements belonging to a domain. Access rights define permissions and prohibitions concerning the execution of methods that belong to an object[3]. *Method classes* can be defined in order to group a particular set of object methods. Consequently, the object oriented approach with role hierarchies, domains and method classes allows extensive use of implicit rights to be made. Explicitly specified *rights* are represented as 7-tuples consisting of

(grantee, protection object, method, predicate, kind of right, grantor, grant flag).

Due to object orientation, access to the Argos services is realised by interface objects which provide methods for administration of the access control system and user interaction. A user is connected to the Argos system via a logon interface that provides a user interface or an administrator interface as the working environment in the case of successful user authentication.

- *User agent*: User related front end process to access the information system via the access control system directly from a HBU. This process runs on a Windows95-PC which is the HBU's IT device. The HBU front end will be further described in chapter 4.

In section 3.1 we will describe the FELIX part of our system which realises *need-to-know security design*, i.e. provides the security information which is necessary to initialise the Argos access control system. Subsequently, in section 3.2 we will describe the CARDS subsystem which realises *context authentication*, i.e. provides the task related information to the access control system which is necessary to evaluate the need-to-know concerning access requests at a particular point in time.

3 NEED-TO-KNOW SUBSYSTEM FUNCTIONALITY

3.1 Security Design

The first component that we have developed to enhance existing access control systems according to our need-to-know access control architecture (see Figure 2) is a security design

---

[3] This refers to the object oriented data model.

system called FELIX[4] which automates the generation of need-to-know security information according to the security design method presented in our earlier papers [Holbein et al. 1996] [Holbein et al. 1995]. Moreover, it allows interactive refinement and extension of the resulting security models. FELIX has been implemented using Ingres Windows4GL (W4GL) because W4GL provides extensive capabilities for human interface design and prototyping, hence it is suitable to implement FELIX as end user tool to be used by security administrators.

FELIX automatically generates security information from transaction based BPMs which correspond to the language/action approach [Winograd 1988]. At this point, we refer to the given literature for a detailed description of transaction based business process modelling. The transaction based BPM are provided by the ActionWorkflow Application Builder system [Action Technologies 1993] [Medina-Mora et al. 1992] [Action Technologies 1993, 1994]. FELIX allows to automatically generate security information which is suitable to initialise role based access control systems according to the individual need-to-know within an organisation's business transactions. Currently, FELIX is restricted to define administrator interface calls for the Argos access control system [Jonscher and Dittrich 1995], i.e. provides means for an individual application of Argos mechanisms. As Argos was developed within a parallel research project there was opportunity to prepare the system for integration within our overall system architecture.

From an overall security design viewpoint FELIX functionality consists of four parts: (1) the definition of security policy principles; (2) the definition of design principles, i.e. generic design constructs according to basic components of transaction based security design; (3) the generation of individual need-to-know security models according to BPM; and (4) manual specification and administration of security information.

The most powerful part of the FELIX functionality (3) provides automatic transformation of ActionWorkflow BPM onto need-to-know security models which can be implemented by Argos mechanisms via Argos administrator interface calls. This is embedded in an interactive environment for security design and administration of security information which also allows a manual refinement and extension as well as additional structuration of the security information that was automatically generated. The result of security design with FELIX are comprehensive security models that can be translated to the corresponding Argos administrator interface calls in order to initialise Argos according to these models.

In the following we give an overview of FELIX functionality. The basic screen of the security design environment is shown in Figure 3. It consists of two main areas: the BPM area and the security information area. The BPM area shows the BPM under consideration. It displays one business transaction construct belonging to a BPM with four lists of subtransactions assigned to its transaction phases and allows movement through all the business transactions belonging to the BPM. However, there are no means for modification of the BPM[5]. A mouse click on one of the subtransactions zooms into this transaction and provides a new arrangement of subtransactions listed around it. A "Task" button for both customer and supplier even allows observation of the task descriptions which belong to the displayed business transaction consisting of the corresponding business transaction activities. The security information area on the other hand shows the results of security design including the security policy principles, the generic design components as well as the need-to-know access rights which result from automatic or manual design activities. In this area a mouse click on a need-to-know access right

---

4 FELIX is a working name not an acronym.

5 Modification of BPMs is not a security design activity. It must be accomplished within business process (re-)engineering where ActionWorkflow provides appropriate tools to be used.

allows modification of its components (see also the description of the *Interactive design* item within the *Design*-menu below). A link between the security information area and the BPM area also provides for trace back from security design to an organisation's business processes. Whenever a mouse click is placed on a need-to-know access right within the security information area, the corresponding BPM will be loaded into the BPM area and the business transaction construct where the access rights has been derived from will be displayed with all its subtransactions listed around it.

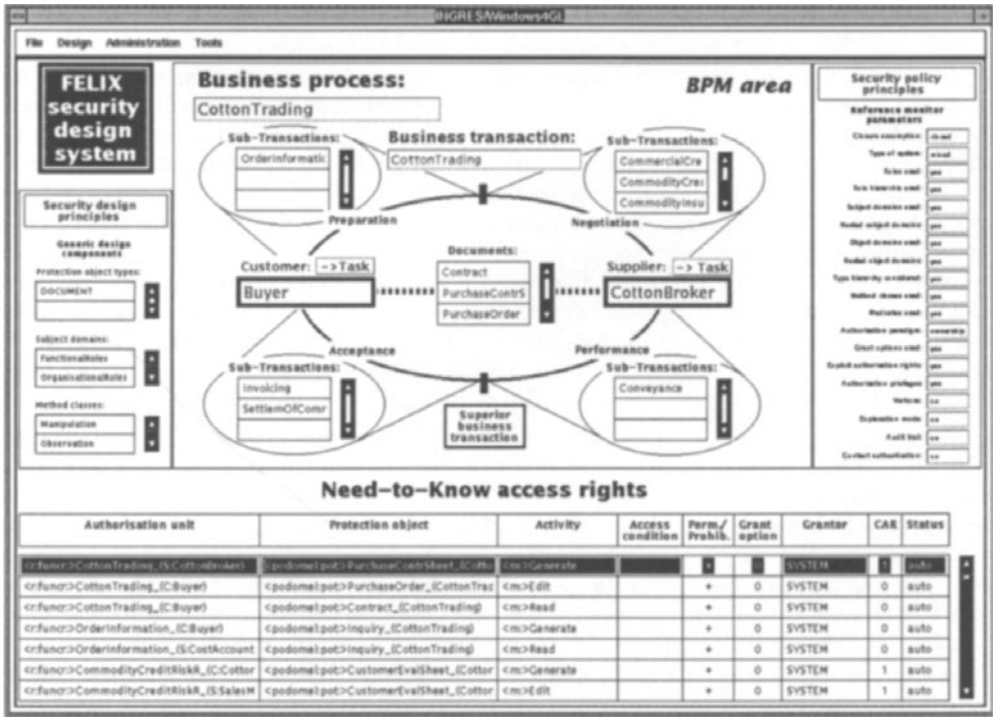

Figure 3: FELIX security design environment

The primary menu structure of FELIX consists of 3 pull down menus that offer file, design and adminstration functions. Unfortunately, the corresponding functionality cannot be explained in detail here. In summary, the *File*-menu basically provides import and export functions. Currently, import of ActionWorkflow BPM as input for security design and export of Argos administrator interface calls for initialisation of the ACS as output of security design (see Figure 2). The *Design*-menu provides the most important functionality that automates security design, i.e. *BPM transformation*, as well as the definition of security policy details, and a CAR-classification (Context Authentication Required) to control discrete CARDS service application (see 3.2). Finally, the *Administration*-menu allows for example, to define user-role associations and to analyse accumulations of rights etc.

3.2 Context Authentication

In this section we describe the CARDS service architecture (Figure 4). CARDS consists of a client part (C-function) that is linked to the access control system as well as a server part (Lotus Notes application) that provides access to the business transaction database (AW transactions) of the ActionWorkflow system. This database provides information on the actual state of an organisation's business transactions. The server part is realised as Notes-client application

because Notes allows ActionWorkflow databases to be accessed in a comfortable way even across different platforms using Microsoft ODBC (Open DataBase Connect).

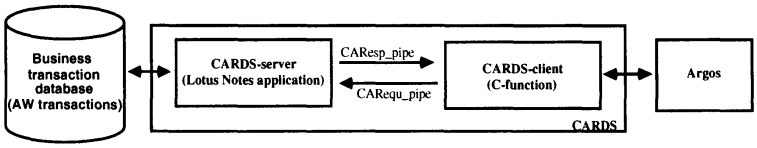

Figure 4: CARDS service architecture

It must be noted that this architecture neither includes an integration within a secure distributed system environment nor provides for high performance. However, this was not a goal of our first implementation which focuses on the context authentication procedure itself as a prototype. Hence, the CARDS-server part is implemented as a Notes-client application running under UNIX and providing an interface to the AWS transactions database running under OS/2. This interface is used to execute the database queries necessary to decide on context authenticity. The queries are specified by the CARDS-client and the query results are evaluated by the CARDS-client again. Communication between the client and server processes is realised by UNIX named-pipes where CARequ_pipe is used to pass database query specifications to the server and CAResp_pipe is used to return the results to the client(s). This architecture allows multiple clients to use the same CARDS server even if the clients are linked to different access control systems in a heterogeneous and distributed environment.

CARDS service access is realised via a CARDS-client which is a C-function that runs under UNIX. It can be called from any ACS that can link this function, exchange a number of parameters and receive a return value. This CARDS() function writes the role name that has been received from the ACS[6] together with the name of the requesting user as a database query specification to the CARequ_pipe. Now the CARDS-client function waits for the requested query result reading the CAResp_pipe. This result is a context certificate that specifies active business transactions and corresponding business process instances as well as the customers associated with these process instances which have been identified in the AW transactions database according to the query specification. The context certificate is signed by the CARDS server using asymmetric encryption techniques and can be verified by the CARDS client using the server's private key[7]. Success or failure of the context authentication finally depends on the following conditions:

- If no context certificate is provided, context authentication has failed (CAF) and a corresponding return code is passed back to the requesting ACS.
- Otherwise, the certificate together with a corresponding return code is passed back to the requesting ACS.

The CARDS-server is realised as Notes application that runs on a 'Lotus Notes workstation for UNIX' (UNIX Notes-client). This Notes workstation allows ActionWorkflow databases to be accessed via a Notes-server, both running on an OS/2 system. Communication between the Notes-client and Notes-server is based on the TCP/IP protocol.

The CARDS-server works as follows: During start-up of the Notes workstation a macro is invoked that calls a Notes function for import of data from the CARequ_pipe. If data comes through the pipe the import function inserts this data into the AWS transactions database using

---

6 Due to their construction these roles include the name of a business transaction which corresponds to workflow names in ActionWorkflow terminology.

7 Public key encryption facilities are provided by Lotus Notes.

a Lotus Notes form that has been defined to contain query specifications. Now, a second macro is started by the database insert event that selects business process instances from the AWS transactions database which includes active business transactions as specified by the inserted data, i.e. a workflow of the specified type with a customer or supplier as defined by the name of the requesting user.

The selected items are used to compose the query result which is a set of records consisting of a business transaction ID, the corresponding business process ID, customer and supplier name of the workflow as well as the customer name of the primary workflow that belongs to the business process. This data is organised in records and signed with the CARDS server's private key. The resulting context certificate is exported to the CAResp_pipe, the exported data as well as the query specification data is optionally deleted from or marked within the database and the macro that reads the CARequ_pipe is restarted to receive the next context authentication request.

4 HOSPITAL APPLICATION SCENARIO

In health care business it is very important to have simultaneous access to different patients data, e.g. health history, patient case data, administrative data etc. Additionally, the patient needs access to various hospital services and features that increase his comfort and safety. Further on, remote monitoring of the patients status (heart, respiration and chemical data) becomes more and more important with the evolution of patient surveillance. Finally, the introduction of Computer Integrated Services (CIS) in hospitals could have an enormous effect to reduce the health cost and enables introducing lean management techniques efficiently and effectively. In this context a project called MobiMed (Privacy and Efficiency of Mobile Medical Systems) was initiated [Fischer et al. 1995]. This project is part of the Swiss Priority Program for information and communication systems and co-operates with a project called EXODUS (EU-Project 00320, Programme ACTS).

The overall objective is to develop and install a Hospital Bed Unit (HBU), which includes a bedside terminal for access to a multimedia information and communication system including overall access to patient data anamnesis, clinical data, administrative and accounting data etc. One reason for that can be found in studies showing that bedside information technology decreased the time nurses spent in documentation activities and increased the time they spent in direct patient care.

In order to implement need-to-know access controls that protect the patients' medical data as described in the previous chapters, the first step is to examine and to model the internal hospital processes. Subsequently, we assume the following scenario: There is a hospital process called General Medicine which includes the phases Registration, Testing, Nursing Cycle, Treatment, Therapy and Discharge of the patient. For a definition of access rights according to the security design method that was implemented as part of our need-to-know access control system (FELIX) we consider these phases as the business transactions which are part of the General Medicine process. Consequently, a Nursing Cycle transaction will be activated as part of each general medicine process if a patient has to stay in the hospital. In this business transaction there is a Nurse who acts as a performer providing the following services to the patient: ordering prescribed medications, administering medications, recording vital signs etc. Service provision requires to have access to the patient's medical history as well as the medical report which is realised via the HBU user interface which has been designed as intuitive as possible, so that health care professionals can get very quick and easy access to patient information.

The following access right for example, results from the business transaction *NursingCycle* and is part of the need-to-know security model that has been automatically derived from the *GeneralMedicine* BPM: **(NursingCycle_(S:Nurse), MedicalHistory, read, , +, 0, SYSTEM, 1, auto)**.

We cannot provide an extensive and complete documentation here because the complete security model would be a vast and rather unreadable set of data consisting of several hundret security records. Therefore, we want to illustrate some practical insights that show benefits and deficiencies of our comprehensive need-to-know access controls within a medical information system. For that reason, we use the need-to-know access right above to explain a corresponding access control trial. The syntax includes the Argos 7-tuple. The access right consists of an authorisation unit representing a combined subject expression with an organisational role *Nurse* resulting from the organisational role of the service provider within the business transaction and a functional role *NursingCycle* according to the business transaction itself. The protection object within the access rights is represented by a protection object expression that corresponds to the *MedicalHistory* information unit which is specified as part of the business transaction. The activity corresponds to the information processing operation called read and was derived from the access type specification within the business transaction. The access right does not include an access condition. It is considered to be a permission (+) and not a prohibition. The receiving authorisation unit (grantee) is not allowed to grant the access rights to other grantees (0). The access right is global, i.e. it was created by a security administrator. This is represented by an artificial user named SYSTEM. Due to the sensitivity of the protection object there is a CAR classification defined for the access right, i.e. *MedicalHistory* has got a CAR classification which implies a CAR flag for each *MedicalHistory* related access right. Finally, the status of the access right is 'auto' because it was automatically generated.

Now we will explain an access control trial that illustrates how success or failure of access requests depends on the current state of the hospital's business transactions. For that reason, we consider a nurse's opportunity for misuse of *MedicalHistory* information. We explain the influence of business process states on the evaluation of an access request that corresponds to the need-to-know access right as explained in the previous section. A person who is assigned to the nurse role is also assigned to all functional roles that correspond to the business transactions where the nurse role occurs. According to our example this is the business transaction called *NursingCycle*. That means, a nurse's need-to-know access rights concerning a patient's medical history are restricted to the *Nursing Cycle* within the *General Medicine* process represented by the corresponding access right above.

We consider the following scenario: the user Petra Müller was assigned to the *Nurse* role as well as the functional role that was defined according to the *NursingCycle* business transaction. Now, we assume Petra Müller has activated the *Nurse* as well as *NursingCycle* roles and sends a request for read access concerning the *MedicalHistory_SamBrown* information object to the access control system. This object is an instance of class *MedicalHistory* where the information owner is a person named Sam Brown. Obviously, the first step of evaluating the access request succeeds because an appropriate role based access right exists. However, due to the CAR-flag within the access right there must be a context authentication for successful evaluation of the access request. In order to explain the possible evaluation results we consider the following four states of the AW manager system concerning the actual business transactions:

1. a *General Medicine* business process instance *GM1* exists, *Nursing Cycle* is the current business transaction within this process and a person named Sam Brown acts as customer within the *General Medicine* process.

2. a *General Medicine* business process instance *GM1* exists and a person named Sam Brown is the customer but *Nursing Cycle* is not the current transaction within this process, i.e. must currently not be accomplished or has already been accomplished.
3. one or more *General Medicine* business process instances *GM1,...,n* exist but Sam Brown is not the customer within one of these processes.
4. no *General Medicine* business process instance exists.

Now it is important to note that although a role based access right exists *the only case where the access request succeeds is number 1 !* This is because the CARDS service provides a context certificate for *Nursing Cycle* and the customer of the business process is a person named Sam Brown which corresponds to the information owner of the requested protection object. In the second case, the access request fails due to a CAF that is provided by CARDS because there is no actual business transaction called *Nursing Cycle* that must be accomplished. The third case fails because even if there is a business transaction *Nursing Cycle,* this transaction is not part of a business process with a customer called Sam Brown. Obviously, the fourth case fails due to a CAF result provided by CARDS.

The access control trial shows that the nurse's opportunity for misuse of personal information can be restricted if the context authentication service is part of the evaluation procedure. The context authentication ensures that the access request succeeds in case of a current need-to-know but not at any time.

5 CONCLUSION AND FURTHER WORK

In this paper, we have introduced a comprehensive need-to-know ACS that is intended to prevent information misuse. However, prevention of information misuse by need-to-know access controls is limited. Our approach reduces the risk of information misuse but it cannot prevent information misuse at all because a user's overall access context is not restricted to his current task within an organisation and therefore, an information misuse risk still remains.

The practical application of our security approach clearly indicated that the security design method cannot be handled manually or in other words, a manual implementation of a need-to-know security model seems to be unfeasible. This is because even single BPM which are comparable to the *General Medicine* process in complexity require tremendous initialisation activities that must be accomplished for implementation of the need-to-know model. However, this security design procedure is necessary to prepare for need-to-know access controls with context authentication. For that reason, a computer based support for security design must be provided to a security administrator in order to apply our approach. The FELIX security design environment appropriately provides for this support by automation of the design process even if a few manual activities are required to complete the overall security model. It is possible to generate a security model consisting of several hundred commands for initialisation of the access control system within a few minutes. Additionally, FELIX allows the generated security model to be efficiently extended, modified and refined.

The CARDS prototype on the other hand allowed us to illustrate the shift from traditional access rights to the notion of rights of disposal concerning intended purposes that can be achieved by a context authentication service. However, the prototype implementation clearly indicated that efficient protocols are necessary in order to achieve system performance that will be suitable for operative use. Even in the case of a discrete and selective application of the context authentication service significant differences for evaluation of access requests will not be acceptable. For that reason, communication between the CARDS client and server

components as well as the server database access procedures must be improved. In a first step, improvement will be possible by direct access to the transaction database of the AW manager, i.e. use of an ActionWorkflow API instead of Lotus Notes facilities. Furthermore, there must be efficient protocols to realise trustworthy communication between the participating communication partners in a distributed system environment.

REFERENCES

Action Technologies I, Ed. (1993). *ActionWorkflow Application Builder User's Guide.* Alameda, CA 94501, USA, Action Technology Incorporation.

Action Technologies I, Ed. (1993, 1994). *ActionWorkflow Analyst User's Guide.* Alameda, CA 94501, USA, Action Technologies Incorporation.

Fischer H-R, Teufel S, Muggli C and Bichsel M (1995) MobiMed - Privacy and Efficiency of Mobile Medical Systems. Project Proposal, Department of Computer Science, University of Zurich.

Holbein R (1996) Secure Information Exchange in Organisations - An Approach for Solving the Information Misuse Problem. Department of Computer Science. Dissertation, University of Zurich.

Holbein R and Teufel S (1995) A Security Service for Role Based Access Controls in Distributed Systems. Presented at the IFIP TC11 Eleventh International Conference on Computer Security IFIP/SEC95, Cape Town, South Africa, 1995.

Holbein R, Teufel S and Bauknecht K (1995) A Formal Security Design Approach for Information Exchange in Organisations. Presented at the IFIP WG11.3 Ninth Annual Working Conference on Database Security, Aug. 1995, Rensselearville, N.Y., USA, 1995.

Holbein R, Teufel S and Bauknecht K (1996) The Use Of Business Process Models For Security Design in Organisations. Presented at the accepted for presentation at IFIP SEC96 TC 11 Twelfth International Conference on Information Security, Samos, Greece, 1996.

IBM (1995) Distributed Security Manager for AIX, Concepts and Planning. IBM Entwicklung Deutschland GmbH, Information Development, Dept. 0446.

Jonscher D and Dittrich K R (1993) A Formal Security Model Based on an Object-Oriented Data Model. Technical Report, Department of Computer Science, University of Zurich.

Jonscher D and Dittrich K R (1995) Argos - A Configurable Access Control Subsystem for Interoperable Environments. Presented at the IFIP WG11.3 Ninth Annual Working Conference on Database Security, Aug. 1995, Rensselearville, N.Y., USA, 1995.

Medina-Mora R, Winograd T, Flores R and Flores F (1992) The Action Workflow Approach to Workflow Management Technology. Presented at the Proceeding of the ACM Conference on Computer Supported Cooperative Work, Toronto, 1992.

Teufel S and Holbein R (1996) Security Aspects of Mobile Medical Systems. Presented at the will be published in Proc. of IFIP TC11 WG11.2 Annual General Meeting on Small System Security, Samos, Greece, 1996.

Winograd T (1988) A Language/Action Perspective on the Design of Cooperative Work. In *Computer Supported Cooperative Work: A Book of Readings* (Greif R, Eds.), pp. 623-653. Morgan Kaufmann Publishers.