Brno University of Technology
Faculty of Electrical Engineering and Computer Science
Department of Computer Science and Engineering

PhD Thesis

Authorization Model for Strongly Distributed Information Systems

---

Ing. Daniel Cvrček

Supervisor: Prof. Ing. Tomáš Hruška, CSc.

Opponents: Prof. dr. ir. W. M. P. van der Aalst
plk. Doc. Ing. Jaroslav Dočkal CSc.
Prof. Ing. Jiří Šafařík, PhD.

Thesis defended on June 1, 2001

Authorization Model for Strongly Distributed Information Systems
© 2001 Daniel Cvrček
ISBN 80-214-1900-8
ISSN 1213-4198

Contents

1 Introduction

Workflow management is a fast developing technology that has gained increasing attention recently. Workflow management systems are very suitable for the business environment because they provide powerful solution for effective administration of business processes, for re-engineering and they allow rapid changes of business processes and environment.

A *workflow management system* (WfMS) is a system that supports specification, execution, coordination and management of workflow processes through the execution of software [12]. Workflow management systems[1] are already used in a wide variety of commercial sectors, e.g. manufacturing, healthcare, banking, insurance companies, service order processing, collaborative software environment, telecommunications, office automation, and many others. Workflow management is a new area and therefore an emerging new area of research. Standardization effort undertaken by the Workflow Management Coalition (WfMC) is also in the very beginning [12, 1].

The application range of the technology is rather wide. What this work wants to introduce is an approach that would ensure secure usage of such big information systems.

2 State of the Art

Despite increasing research efforts, commercial products by software vendors and standardization efforts by Workflow Management Coalition, security aspects in WfMS have not been given enough attention.

In the following, we elaborate the problems encountered in enforcing security in strongly distributed information systems.

Strongly distributed systems are very complex for security management. First condition for enforcing security is to split the distributed system into *homogeneous* (from the security point of view) parts that are centrally administered. We shall call them *autonomous information system* or *s-node*. Distributed system is then composed form a number of s-nodes that have to securely cooperate. To ensure such a secure cooperation we have to solve (beside others) authorization control. One may found several areas of authorization control to solve.

[1] Throughout the work, we also use the notion of strongly distributed system as an abstraction of WfMS but it may be understood as equivalent.

1. Access control in autonomous system - This comprises access control to resources local in the s-node and it is solved by access control model implemented in the local platform (operating system, database management system, ...) the s-node uses. Each autonomous system may have different access control model. And even in the case access models are the same, security policies used on s-nodes may differ and may therefore be incompatible.

2. Global administration of system - Distributed system has to solve, somehow, problems with heterogeneity of its s-nodes and enforce uniform administration of security properties. This demands some sort of backbone, some basis that may be used as referential by all s-nodes.

3. Flow control - We are talking about systems that allow (space) distributivity of computational tasks. Those tasks may use data with different sensitivity, stored on many s-nodes. There has to be a common template, common rules for data flow control, some *reference monitor* [2].

The work [13] uses notions discretionary and mandatory security for WfMS. We do not think that this is the correct partitioning. Discretionary access control means that access control is enforced by the owner of the resource, but it is not what the security in WfMS is about. On the other side, we may discern several layers in security of a distributed system.

1. Access control to resources on s-node - It is the lowest layer. We may use discretionary access control or mandatory access control or anything else. Not users, but tasks (that are invoked by users or other tasks) access resources according to workflows' definitions. The principal problem is to replace a task with the proper user on the s-node.

2. Access control to resources in workflow - It is higher layer. We have to define authorizations in workflow in some common way. One may say there is no difference between this and the previous item, but the opposite is true. The difference is in the way access rights are specified (s-node vs. distributed system). And the conversion between those two ways is an important issue for security in heterogeneous systems.

3. Access control to workflows - It comprises management of activation of computational tasks by users and other tasks. This is again more abstract view on access control.

4. Flow control of workflows - This is in the same layer as access control in workflows. We may say that access control and flow control are two sides of the same coin.

You can see that *discretionary* is able to satisfy security requirements only in the lowest level of authorization control. An axiomatic access control has to be on the other side used for access control to workflows (items 2-4). We may use mandatory access control, or more relaxed policies. The WfMS security is very often connected with role-based access control [4, 11, 17, 15, 5, 6, 16].

3 Objectives

The target of the thesis is to develop system ensuring secure execution of long-timed computational tasks performed on several s-nodes. A formal model that will settle rules for secure cooperation of various access control models is necessary condition. Design of such a formal model covers number of research issues:

1. Definition of a methodology that sharply discerns security problems that shall be solved on particular layers of security model of distributed system. The layers shall be specified correctly.

2. A formal security model for task-based authorizations does not exist. Such a theoretical model is necessary to provide a basis for formal proofs that allow to establish higher confidence in the correctness of the workflow.

3. A formal framework that covers all layers of security in the heterogeneous distributed system's environment does not exist. We may identify the following areas of problems according to layers of security abstraction.

   (a) Access control to resources in computational tasks and especially its cooperation with underlying access control models is not solved. It is necessary to ensure secure transition of access privileges from global definitions in computational tasks into definitions usable in s-nodes' access control systems and vice versa. This transition must have properties that prevent obtaining access rights by unauthorized subjects (users, tasks), but also ensure successful execution of tasks.

   (b) Control of information-flow and authorization-flow has to be maintained in such a way that allows enforcement of mandatory access control when necessary. This control is based on classification of resources and subjects. The constrained information flow must ensure at least one of:

i. Confidentiality - the task execution does not give raise to direct or indirect illegal disclosure of sensitive information.

ii. Integrity - the task execution does not allow any unauthorized changes of data.

The controls still have to ensure availability of resources that are necessary for execution of tasks and should protect computational tasks from delays and starvation.

We have got three basic properties of secure data access: confidentiality, integrity and availability. It is natural that availability goes down any time we increase control of confidentiality or integrity of resources. Further, confidentiality and integrity are opposites (in some sense) and their demands go against each other. It means that any independent tightening of requirements on confidentiality or integrity worsens availability.

This is one of the key issues of mandatory access control. The control is so tight that its usage in commercial environment is ineffective or even impossible. It has been the reason for role-based access control to appear.

(c) Access rights for activation of workflows (computational tasks) have to be solved in such a manner that allows effective access control and tasks' distribution throughout the distributed system.

4. Construction of new theories that use definition of authorization model according to the previous paragraphs.

To succeed, we have to solve minimum subset of the above mentioned areas. We have to solve items 1 and 2 completely. 3a) is going to be solved generally by introducing possible mechanism. Item 3b) has to be solved at least for confidentiality or integrity and 3c) must be solved fully, of course. We shall introduce categorization and use it when appropriate. Last chapter of the thesis defines *secure inheritance* of processes that may be understood as a representation of item 4.

4 Structure of the Work

Generally, this dissertation identifies specific requirements for incorporating security into distributed information systems and proposes basic security model that allows synchronization of workflow execution and authorization flow.

There is proposed classification of security layers that allows easier treatment of security problems, the model is defined and incorporated into formalism for communicating processes and idea of inheritance of processes is formalized by previously created theory.

The text of the thesis is divided into nine chapters. Two chapters contain problem statement and short survey of security in information and database systems. Following chapter presents a brief introduction into *process algebra* - CCS. CCS (Calculus of Communicating Processes) [14, 10] has been designed for modeling communicating processes.

Rest of the thesis contains original contribution to the field of security in strongly distributed systems. The theory uses CCS and ideas of inheritance of processes. The CCS theory and its description is strongly used in definition of Secure CCS, where the original calculus is modified to satisfy security requirements. Next chapter describes general architecture of authorization system for strongly distributed information systems. The architecture is the cornerstone of the subsequent construction. The basic structure of authorization system is defined and it is followed by description of basic communication and cooperation among layers.

One chapter introduces Active authorization model (AAM). This model is the basis for authorization system. The original idea has been to use this model as is, but it showed up that a formalism for tasks' definitions is needed to demonstrate applicability of AAM. The original idea is the reason why e.g. relations of authorization steps are defined. Three sections of the chapter are denoted to definition of distributed system, basic concepts of authorization model and complete formal AAM definition itself.

Next chapter (Incorporating Security into CCS) is the first chapter that addresses security in CCS. It consists from three sections. "Refinement of AAM" that makes minor changes in AAM necessary for successful incorporation into CCS. It introduces definitions of access and flow control, specifies their manipulation and defines classification/categorization of system elements. Next section introduces forms of authorization information for states and defines operator that enables *joining* of states. Last section defines properties of secure distributed system and secure tasks (and workflows).

Chapter "Secure CCS" goes through specification of CCS and makes appropriate changes to ensure security requirements stated in previous chapters. The aim was to introduce as little changes in CCS as possible. The resulting *Secure CCS* ensures properties of secure distributed system while preserving simplicity and useful properties of the original specification. The most important are notions of bisimilarity that had to be changed, but the original ideas

stay preserved.

The last part of the work tries to incorporate ideas of Van den Aalst and Basten about inheritance of processes into the frame developed in the previous chapters. The thesis concludes with results of the work and possible directions for future research.

It is very difficult to depict part of the work because it is very compact and its parts are chained together. We have decided to resume some of basic notions to illustrate the approach used in the work. Following pages contain selection of some important problems solved in the thesis to show construction of the theory. We should start with some basics that are needed for definition of secure system. Some notions of Secure CCS are resumed and inheritance of secure processes concludes the chapter.

4.1 AAM - Selection

Original section is very compact and contains complete specification of Active authorization model. At this place, we are to introduce just a set of selected notions.

Definition 4.1. (Authorization set) Let us O_i be a set of all resources, S_i be a set of all subjects and A_i be a set of all access modes on an s-node. If there is defined $k \in \mathbb{N}$ then $U_i = \{u \mid u \in \mathbb{N} \vee u < k\}$ is a set of all possible repeating counts. Then the set of all authorizations P_i is defined as

$$P_i \subseteq S_i \times O_i \times A_i \times U_i$$

For that holds:

$$\forall\, p = (s, o, a, u) \in P_i : u > 0,\ \exists p' = (s, o, a, u - 1) \in P_i$$

■

This is the first and the last time when subjects are directly associated with authorizations in AAM. The reason to define authorization sets is a need to express concrete privileges on s-nodes and to have a basis for deriving more abstract notions for authorizations. We are going to use association *task – privilege* from now on. Because users are not needed for that specification, there are also introduced authorization templates. There are two forms of them - specific (for particular objects) and abstract (for types of objects).

Definition 4.2. (Authorization application) We define a function Λ that expresses application of an authorization. There are the same rules for authorizations, as well as for both types of authorization templates. Domains of the function are therefore P_i, Π^A and Π_i^S.

1. $\Lambda : P \mapsto P$,
   $p = (s, o, a, u) \in P \wedge u > 0$ then $\Lambda(s, o, a, u) = (s, o, a, u - 1)$
2. $\Lambda : \Pi^A \mapsto \Pi^A$
   $\pi^A = (c, a, u) \in \Pi^A \wedge u > 0$ then $\Lambda(c, a, u) = (c, a, u - 1)$
3. $\Lambda : \Pi^S \mapsto \Pi^S$
   $\pi^S = (o, a, u) \in \Pi^S \wedge u > 0$ then $\Lambda(o, a, u) = (o, a, u - 1)$ ■

We define initial authorizations to allow definition of conditions that subjects initializing tasks on the particular s-node have to satisfy.

Definition 4.3. (Initial authorizations) Let us assume that P_i is a set of authorizations and Π^A is the set of respective abstract authorization templates. Then let P_I^t be the set of initial authorizations (for the task step t).

$$P_I^t \subseteq \Pi^A$$

is the set of initial authorizations. ■

We should notice at least two facts. There is no constrain for sets of objects. It means that we may want a subject working on one s-node to have authorizations for objects on another s-node. The second fact is that there has to be at least one subject (in the system) that satisfies the initial authorizations. This condition is necessary for execution of the associated task. All subjects that posses the initial authorizations are called initiators.

Definition 4.4. (Initiators) Let Π^A be the set of authorization templates, S_i be the set of subjects on s-node $\mathcal{S}_i$ and P_I^t be the set of initial authorizations chosen over Π^A and $S_0^{t,i} \subseteq S_i$. We define function Φ_i:

$$\Phi_i(S_i, P_I^t) \mapsto S_0^{t,i}$$

that returns a set of subjects $S_0^{t,i}$ on s-node $\mathcal{S}_i$ those privileges form superset to P_I^t. The set $S_0^{t,i}$ is then called *initiators* for P_I^t (on s-node $\mathcal{S}_i$). ■

Basic authorization data structure is authorization unit. It is associated with task step - part of task that is executable by one subject in *one step*. We may say it is atomic for users.

Definition 4.5. (Authorization unit revisited) We have a set of abstract authorization templates Π^A. Let P_I^t be the proper set of initial authorizations and $P_E^t = (P_E^{A,t}, P_E^{F,t}) \subseteq \Pi^A \times \Pi^A$ be enabled authorizations consisting of the set of access control ($P^{A,t}$) and the set of flow control ($P^{F,t}$) authorizations. Further, let $S_e^t \subseteq S_0^t$ be a set of executors of the authorization unit:

$$S_e^t = \{s_e^t | \, \exists i, s_e^t \in S_i \wedge s_e^t \in \Phi_i(S_i, P_I^t)\}$$

Each task may be repeated several times, so we define number of initializations of the associated authorization unit $n \in \mathbb{N}$. The authorization unit ω_i is then defined as a 4-tuple:

$$\omega_i = (P_I^t, P_E^t, n, S_e^t)$$

The most important are the first three elements of ω_i. ■

Assuming execution of any authorization unit we are not interested in application of particular enabled privileges. This application is done according to Def 4.2. We shall denote authorization unit as $\omega_{i,k}^l$ after l-th application of enabled privileges.

Definition 4.6. (Validity of authorization unit) Assume, we have the authorization unit ω_i in the k-th state $\omega_{i,k} = (P_I^t, P_E^t, S_{e,k}^t, n_k)$. One says that ω_i is invalid if and only if $n_k = 0$. Otherwise ($n > 0$), the authorization unit is valid. ■

4.2 Categorization and Classification

Each process that creates new data, applies all the data it receives (through communication as well as through direct access to s-node resources). Categorization of the output must respect properties of all input data. We have to go even further and say that categorization of all data that are changed, created or stored on an s-node should look at categorization of all input data.

We have used the word *respect* input data. Before we try to formalize this, we have to define categorization. The following definition introduces exact specification of categorization (classification). The basic idea comes from BLP model [2] and lattice model for flow control [7, 8, 9].

Definition 4.7. (Classification) Each object has got assigned security classification that is defined on the set of data categories DC and on the set of security classes SC, $C = SC \times DC$. Classification is a couple $c_i = (sc_i, dc_i)$.

Each security class is an element of ordered set $SC = \{sc_1, \ldots, sc_k\}$ according to operation $\leq$ $(sc_1 \leq sc_2 \leq \cdots \leq sc_k)$.

Each category $dc_i \subseteq DC$ and categories constitute a partially ordered set of sets with minimal element $\emptyset$, maximum element DC and operation $\subseteq$.

Classifications constitute a lattice with minimum element $(sc_1, \emptyset)$ and maximum element (sc_k, DC). Elements are partially ordered:

$$(sc_1, dc_1) \leq (sc_2, dc_2) \Leftrightarrow (sc_1 \leq sc_2) \wedge (dc_1 \subseteq dc_2)$$

The lowest upper bound $\triangle$ is defined as

$$(sc_1, dc_1) \triangle (sc_2, dc_2) = (max(sc_1, sc_2), dc_1 \cup dc_2)$$

and greatest lower bound $\triangledown$ as

$$(sc_1, dc_1) \triangledown (sc_2, dc_2) = (min(sc_1, sc_2), dc_1 \cap dc_2)$$

Each element c_i of security classification is called a *classification class*. ■

Definition 4.8. (Categorization) Assume, we have a classification C defined on the set of data categories DC and on the set of security classes SC. Assume that $SC = \{sc_1\}$. In other words, all data has got the same *security class*. We then call C a *categorization*. Each element of a given categorization is called *categorization class*. ■

When respecting security properties of input data, the classification of output data c is *join of classification classes* of all input data $c_1, \ldots, c_n$. It is defined as $c = c_1 \triangle \cdots \triangle c_n$.

Definition 4.9. (State classification) There is a function Γ that assigns a classification class to each state. Let's say that $\mathcal{P}$ is the set of all states, $C = \{c_1, c_2, \ldots\}$ is the classification ($(sc_1, \emptyset)$ is the minimum element of C), p_i enabled authorizations and ω_j are authorization units.

$$p_i = ((o_{i,A},\, a_{i,A},\, u_{i,A}),\, (c_{i,F},\, a_{i,F},\, u_{i_F}))$$
$$\omega_j = (P_I^t,\, (\{(o_{i,\,A},\, a_{i,A},\, u_{i,\,A})\}_{i \in I},\, \{(c_{j,\,F},\, a_{j,F},\, u_{j,\,F})\}_{i \in J}), S_e^t,\, n)$$

$\Gamma : \mathcal{P} \rightarrow C$ is defined as:

$$\Gamma(P_{p_i}) = \Theta(o_{i,A}) \vartriangle c_{i,F}$$
$$\Gamma(P_{\omega_j}) = \triangle_{i \in I}\ \Theta(o_{i,A})\ \ \triangle_{i \in J}\ c_{j,F} \vartriangle (sc_1, \emptyset)$$
$$\Gamma(P_{P_I}) = (sc_1, \emptyset)$$
$$\Gamma(P_{\sigma_i \cup \sigma_j}) = \Gamma(P_{1,\sigma_i}) \vartriangle \Gamma(P_{2,\sigma_j})$$

■

4.3 Definition Of Secure System

The properties, whose compliance ensures security of a given system are stated in this section. We also have to formalize them to be used as a basis for proving security of the proposed authorization system.

Security properties are divided into two sections, access control and flow control. There may be systems those tasks do not allow communication or communication is not important feature. Such systems define categorization of data, but all users are allowed (generally) to access all data sent among task steps of workflows and tasks are used only to define their job. In that case only access control is what we are interested in. On the other side, flow control makes the system generally secure and should enable definition of rules that correspond with principles of mandatory access control.

We state properties necessary for systems to be called secure. They are divided into two groups, related to access control and flow control.

4.3.1 Access Control of Secure System

We have already several times mentioned that authorization unit is the basic information to define security properties of tasks. This is confirmed in the following items that use the notion to define properties of secure system.

1. Each workflow definition defines maximum number of initializations for all authorization units it contains.

2. Authorization unit defines, through enabled authorizations, set of authorizations that may be granted to a subject (a task or a user).

3. A subject may be granted just the minimum set of privileges necessary to execute the authorization unit. This set is specified by enabled authorizations.

4. A subject is granted authorizations defined in an authorization unit only for the time he executes it. The necessary condition for the execution is satisfaction of initial authorizations defined in the executed authorization unit.

5. A subject may execute only one task at a time.

6. Application of any authorization from enabled authorizations causes decreasing of available initializations of given authorization unit[2].

As you can see, properties of access control correspond with properties of Active authorization system. Property (1) is implied by Def 4.5 as well as Property (2). Property (3) is expressed in definition of $\triangle$ operator used for abstraction of authorization information. Property (6) is enforced by definition of authorization application - Def 4.2 and Def 4.6 (validity of authorization unit). Requirements (4) and (5) are directed towards properties of Workflow engine.

4.3.2 Flow Control of Secure System

To define flow control we need classification (Def 4.7) and its elements, classification classes, or categorization (Def 4.8).

1. Each data in the system is assigned a classification class. This class is defined as static or dynamic according to type of the given resource. The static class does not change during the data existence. The dynamic class is affected by flow control (by classes of all input data).

2. There is defined a *flow relation* on pairs of classification classes. Given two classes c_1 and c_2, the relation $c_1 \rightarrow c_2$ is valid if, and only if, information in c_1 is allowed to flow into c_2 ($c_1 \leq c_2$, according to Def 4.7).

3. There is defined operation $\triangle$ for *classification class combination* that defines classification class resulting from interference of two classification classes: $c_1 \triangle c_2 = c_3$.

4. There does not exist a sequence of operations that violates the flow relation. If a value $f(a_1, a_2, \ldots, a_n)$ flows into an object with class b, then the relation $a_1 \triangle a_2 \triangle \ldots \triangle a_n \rightarrow b$ must be satisfied.

[2]It holds even for fail of the unit execution. We understand that there may be different opinions on this question. We can imagine rules that define number of possible initialization with fail, but for simplicity we use this rule.

Item (1) specifies property of Workflow engine and is necessary for all other flow control properties. Item (2) states when a data flow is correct. Next property shall be used for classification of output ports (output data) and item (4) puts new requirement on subsequent task steps.

Access control is stated by AAM. When showing security of a new task, all we need to do is to show compliance with properties of flow control.

4.4 Processes in Secure CCS

The authorization model needs to make difference (in contrary to CCS) among three basic abstraction levels for process modeling.

1. E-complete - Atomic processes on s-node level. Processes are *complete* to perform some atomic action defined by system (e.g. operation system) that is installed on a particular s-node. (An example may be to read a given file or to write a record in a database.)

2. A-complete - Atomic processes on the authorization system level. They are *complete* to be granted basic set of authorizations (expressed in an authorization unit by enabled authorizations) and to perform an atomic action defined in the context of the distributed system. (We may state examples like changing balance of an account or filing an insurance claim.)

3. W-complete - Composite tasks (workflows) that may be initiated by users. Those tasks represent the layer of a distributed system that is managed by users (users may decide whether initiate a workflow). With a-complete processes, users may decide when, but necessity to run a process is stated by the system.

Processes (or *states* in the process calculus) on the finest resolution, *a-complete processes*, change authorization unit ω_i by removing elements from the set of enabled authorizations of the authorization unit. Assume that $P^t_{E,i} = \{p_1, \ldots, p_j, \ldots, p_k\}$ then $P^t_{E,i+1} = \{p_1, \ldots, p'_j, \ldots, p_n\}$, where $p'_j = \Lambda(p_j)$ is result of application p_j during execution of associated e-complete process.

$P^t_{E,i}, P^t_{E,i+1}$ are enabled authorizations from consecutive states of an authorization unit and p_j is the authorization applied in the former state. We are not interested in initial authorizations on this level of abstraction.

The second level of abstraction works with execution of authorization units. This level takes P_E^t as a primitive entity. The set of initial authorizations is used here, but no users are allowed to initiate the task on this level of abstraction by themselves. Authorization units are parts of task steps of a workflow and the distributed system determines whether (and sometimes when) to execute them and searches for appropriate subjects when needed.

The highest level of abstraction is represented by tasks that may be initiated by users. To initiate a task, we need to define conditions for it - initial authorizations. Those authorizations are defined accordingly to the needs of application environment. Initial authorizations defined for particular steps t_j of the workflow are independent on the initial authorization of the workflow.

We do not need to assume sets of enabled authorizations on this abstraction level. The same may hold for the number of possible initializations n.

4.5 Secure CCS - Action and State

We have discussed some basic properties that have to be satisfied for modeling of access and flow control. Let us start with actions that represent communication[3] between processes. Assume, we have an infinite set $\mathcal{A}$ of names and we use $a, b, c, \ldots \in \mathcal{A}$ as names. Those names are used for communication between agents.

Definition 4.10. (Names) Assume, there is a set of names $\mathcal{A}$ and data classification $C = \{c_1, c_2, \ldots, c_n\}$ defined on DC and SC. Each element $a \in \mathcal{A}$ is assigned a classification $c_i \in C$. We denote a name as a_{c_i}. ■

Definition 4.11. (Co-names) Assume, there is a set of names $\mathcal{A}$. We denote by $\bar{\mathcal{A}}$ the set of co-names. We say that $\bar{a}_{c_i} \in \bar{\mathcal{A}}$ is complement of $a_{c_i} \in \mathcal{A}$. The classification class c_i of $\bar{a}$ is the same as the classification class of a. ■

Definition 4.12. (Labels) Labels is union of sets of names and co-names. $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$. ■

States in the system are represented as agents (agent constants, agent expressions, ...). We have said that each state possesses its security information.

[3]We should talk about synchronization because we are going to work with basic calculus that does not offer value passing. We have got two reasons for that. Basic calculus is simpler for development of theory and CCS offers simple way for reducing full calculus into the basic one.

The type of information depends on the abstraction the state represents (e-complete, a-complete or w-complete process). Each state possesses generally a set of tuples of static security information (they are used for access control). Each tuple may contain enabled authorizations $P_E^t = (P_{E,t}^A, P_{E,t}^F)$ where $P_E^{A,t} = \{p_{e,t}^A\} \vee \emptyset$ and with authorization unit ω_t or a set of initial authorizations P_I^t. The allowed forms of authorization information are:

$a_i = ((p_{e,i}^A, P_{E,i}^F),\ \emptyset,\ \emptyset,\ (p_{e,i}^A, P_{E,i}^F),\ \emptyset)$ for e-complete process
$a_i = ((\emptyset, P_{E,i}^{F+}),\ \omega,\ \emptyset,\ P_E^+,\ P_I^+)$ for a-complete process
$a_i = ((\emptyset, P_{E,i}^{F+}),\ \emptyset,\ P_I^i,\ P_E^+,\ P_I^+)$ for w-complete process
$a_i = ((\emptyset, P_{E,i}^{F+}),\ \emptyset,\ \emptyset,\ P_E^+,\ P_I^+)$ for all other processes

Where P_E^+ and P_I^+ are sets of *counted authorizations* that are *counted* from authorizations of all states the given state is composed of. Those are used for determination of proper authorization sets in more abstract states.

To shorten notation of authorization information, we denote agents with σ_t that represent all authorization information assigned to the state. We may write a transition as:

$$P_{\sigma_i} \xrightarrow{\ell_c} Q_{\sigma_j}$$

P_{σ_i} and Q_{σ_j} are agents, $\ell_c \in \mathcal{L}$ is a label.

We still need to introduce one last action. It is action that arises each time when b_{c_i} and $\overline{b}_{c_i}$ (complementary actions) are invoked simultaneously. This action is internal for a composite agent, i.e. the same action arises from any pair of complementary actions.

Definition 4.13. (Perfect actions) The action that denotes simultaneous invocation of any complementary actions $a_c, \overline{a}_c$ is called *perfect* or *completed action* and is denoted τ_c. This action has got assigned the same classification as actions it represents - c.
τ_* denotes set of all perfect actions. ■

All perfect actions have the same visible effects - none. We need to preserve classification of original actions and that is the only difference among perfect actions.

With τ_* actions, we may define set of all actions $Act = \mathcal{L} \cup \tau_*$.

Perfect actions play very important role because they allow us to ignore all internal (perfect) actions of composite systems. This is enabled by the fact that τ_* actions do not represent any potential communication and they are therefore not directly observable. But because each τ_c represents two actions

it is reasonable to apply restriction on the composite system. It should not use the *hidden* actions.

Definition 4.14. (Port) Each state has ports denoted by names from $\mathcal{L}$. The ports denoted by any $a_{c_i} \in \mathcal{A}$ are able to receive (input ports), ports denoted by $\overline{a}_{c_i} \in \overline{\mathcal{A}}$ are able to transmit (output ports).
To define an action a_{c_i} from agent P_{σ_i} to Q_{σ_j}, there must be defined ports a_{c_i} and $\overline{a}_{c_i}$ in P_{σ_i} and Q_{σ_j}, respectively. ■

Talking about ports, they have to satisfy security properties defined for secure system. It is especially item 2 and item 4 of flow control.

Definition 4.15. (Secure port) Secure port is a port that satisfies conditions for secure system.

1. *Input port* - classification class of the input port $a_{c_i} \in \mathcal{A}$ is the same as of the corresponding output port $\overline{a}_{c_j} \in \mathcal{A}$.

   $$c_i = c_j$$

2. *Output port* - classification class of each output port $\overline{a}_{i,c_i} \in \mathcal{A}$ for the given state P_σ is equal or greater than *combination* of all input ports $a_{j,c_j} \in \mathcal{A}, \forall j \in J$ of the state P_σ and classification of all static resources accessed in P_σ.

   $$\forall j \in J : \Gamma(P_\sigma) \leq c_j$$

Input ports in the definition are an abstraction that also represents places to receive all *data flows* from local resources (access control). ■

Lemma 4.1. *Secure port preserves security of distributed system.*

Proof. We use Def 4.14 declaring that action and input and output ports it connects, are all assigned with the same classification class. We are talking about data flow, so we have to prove fulfillment of property 2 for input ports and property 4 for output ports (page 11).

1. A transition represents a data flow, so *flow relation* $\rightarrow$ must hold. Assume, we have a transition $P_{\sigma_p} \overset{a_{c_a}}{\rightarrow} Q_{\sigma_q}$ that connects ports a_{c_a} and $\overline{a}_{c_a}$. It implies flow relation $c_a \rightarrow c_a$. Because $c_a \leq c_a$ by Def 4.7, the flow relation is valid and property is satisfied.

2. A state represents operations and therefore inference of classification classes that are represented by classification classes of all input ports $a_{i \in I}$ of the given state. The relation is

$$\triangle_{i \in I}\ c_i \rightarrow c_j \Leftrightarrow c'_j \rightarrow c_j$$

According to Def 4.15 (2), $c'_j \leq c_j$. It means that by Def 4.7, the flow relation is valid and property is satisfied.

□

Definition 4.16. (Secure state) The state that contains only secure ports and that satisfies all relevant requirements of access control for secure system is called *secure state*. ■

Remember definition of state classification (Def 4.9). Now, when ports have been introduced, we could redefine classification of state using its ports (flow control) and inner properties (access control). The resulting classification of the state would be combination of input ports' classification classes (it covers also classification of resources, the state is allowed to access).

Definition 4.17. (State classification in CCS) There is a function Γ that assigns a classification class to each state. Let's say that $\mathcal{P}$ is the set of all states, $C = \{c_1, c_2, \ldots\}$ is the classification (let $(sc_1, \emptyset)$ be the minimum element of C) and P_σ is a secure state and $\{a_{1,c_1}, \ldots, a_{n,c_n}\}$ is the set of all input ports in P_σ. $\Gamma : \mathcal{P} \rightarrow C$ is defined as:

$$\Gamma(P_\sigma) = \triangle_{i=1..n}\ c_i\ \triangle\ (sc_1, \emptyset)$$

In case of no input ports in P_σ $(n = 0)$, then holds $\Gamma(P_\sigma) = (sc_1, \emptyset)$. ■

This is very short selection of original chapter. Frankly, basic notions from the beginning of the chapter have been chosen. The key notions of bisimilarity would need much more place to introduce, so we present just secure bisimilarity in the following section.

4.6 Inheritance of Processes

Inheritance of secure processes is naturally more complicated than inheritance without considering security. The way we are going to follow is partially

determined by van den Aalst and Basten. We are to introduce new set(s) of axioms that allow us to extend equivalence of processes.

When defining inheritance in CCS, we firstly have to introduce new axioms that are definitely distinct from the original ones defined in ACP. We are going to introduce two basic forms of inheritance. They are based on ignoring effects of actions (actions are replaced by silent actions) and on deferring some actions (appropriate branches are excepted). We have some advantage because CCS contains restriction operator that can be used to define one form of inheritance and there is also a relabelling function that can be used for the latter one.

But we have to go further, there is the second problem - equality of secure processes. Let us assume the following example. We have definition of a process - task. We create a subprocess (from now on, the notion is used for inherited process) by adding new branches, new parallel states and actions. In the moment we start to hide those new actions (or their effects) we can not do the same with all the related security information. The result is that we obtain two equal processes from the functional point of view, but security properties of them are different.

To solve problem with equality of processes, all possibilities were explored: some changes in axioms, changes in definitions of inheritance, ignoring security properties to some extent. The solution we have finally chosen lies in minor changes in inheritance definitions, but primarily in introducing new bisimulation (it is more exactly preorder).

4.6.1 Axioms

We are going to extend set of axioms for finite state agents. Secure CCS contains two sets of axioms for finite state agents that express monoid laws and τ-laws of CCS theory.

Axioms $\mathcal{A}_1$

A1 $P_{\sigma_p} + Q_{\sigma_q} =^s Q_{\sigma_q} + P_{\sigma_p}$
A2 $P_{\sigma_p} + (Q_{\sigma_q} + R_{\sigma_r}) =^s (P_{\sigma_p} + Q_{\sigma_q}) + R_{\sigma_r}$
A3 $P_{\sigma_p} + P_{\sigma_p} =^s P_{\sigma_p}$
A4 $P_{\sigma_p} + \mathbf{0} =^s P_{\sigma_p}$

The second axiom system contains also the τ laws:

Axioms $\mathcal{A}_2$

A1-A4
A5 $\alpha_{c_\alpha}.\tau_{c_1}.P_{\sigma_p} =^s \alpha_{c_\alpha}.\varepsilon_{c_1}.P_{\sigma_p} =^s \alpha_{c_\alpha}.P_{\sigma_p}$
A6 $P_{\sigma_p} + \tau_{c_1}.P_{\sigma_p} =^s \tau_{c_1}.P_{\sigma_p}$
A7 $\alpha_{c_\alpha}.(P_{\sigma_p} + \tau_{c_1}.Q_{\sigma_q}) + \alpha_{c_\alpha}.Q_{\sigma_q} =^s \alpha_{c_\alpha}.(P_{\sigma_p} + \tau_{c_1}.Q_{\sigma_q})$

The basis shall be set of axioms $\mathcal{A}_2$. We need τ actions to allow abstraction. This is because of the definition of inheritance that is partially based on *ignoring* actions new in subprocess.

First task is to find a way to express encapsulation and abstraction. Basten and Van den Aalst use ACP [3] process algebra and they introduce two new operators and use special actions τ and δ for results. We know τ action in the process calculus, but we do not use δ actions (inaction). We assume that it is possible to use inactive agent $\mathbf{0}$ instead.

We have two basic types of inheritance: protocol and projection. We therefore define two new sets of axioms for each type and the third one for their composition. We obtain axioms for protocol inheritance, projection inheritance and the last set contains axioms for both types of inheritance.

Axioms $\mathcal{A}_3$ $L \subseteq Act$

A1-A7

D1 $\alpha_{c_\alpha} \notin L \cup \overline{L} \Rightarrow (\alpha_{c_\alpha}.P_{\sigma_p})\backslash L =^s \alpha_{c_\alpha}.P_{\sigma_p}\backslash L$

D2 $\alpha_{c_\alpha} \in L \cup \overline{L} \Rightarrow (\alpha_{c_\alpha}.P_{\sigma_p})\backslash L =^s \mathbf{0}$

D3 $(P_{\sigma_p} + Q_{\sigma_q})\backslash L =^s P_{\sigma_p}\backslash L + Q_{\sigma_q}\backslash L$

For the projection inheritance, we use a relabelling function. We may call it τ_L, where $L \cup \overline{L}$ is the set of labels that are to be hidden.

Axioms $\mathcal{A}_4$ $L \subseteq Act$

A1-A7

I1 $\alpha_{c_\alpha} \notin L \cup \overline{L} \Rightarrow \tau_L(\alpha_{c_\alpha}) =^s \alpha_{c_\alpha}$

I2 $\alpha_{c_\alpha} \in L \cup \overline{L} \Rightarrow \tau_L(\alpha_{c_\alpha}) =^s \tau_{c_\alpha}$

I3 $(P_{\sigma_p} + Q_{\sigma_q})[\tau_L] =^s P_{\sigma_p}[\tau_L] + Q_{\sigma_q}[\tau_L]$

I4 $(\alpha_{c_\alpha}.P_{\sigma_p})[\tau_L] =^s \tau_L(\alpha_{c_\alpha}).P_{\sigma_p}[\tau_L]$

Axioms $\mathcal{A}_5$

A1-A7, D1-D3, I1-I4

4.6.2 Definition of Inheritance

We are ready to present new definition of observation bisimulation that allows a little different treatment of authorization information. The definition of secure equality uses previously defined notion of observation equivalence. Both relations are equivalence relations. We need to introduce new notions that allow to define inheritance with one relaxation. Authorization information of states and actions in a subprocess is equal or *greater* to respective authorization information in the base process. The resulting relation is not equivalence because it lacks symmetry of security information, but preorder. We shall call the new relation *inheritance bisimulation*.

Definition 4.18. (Inheritance bisimulation) A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over secure agents is an *inheritance bisimulation* if $(P_{\sigma_p}, Q_{\sigma_q}) \in \mathcal{S}$ implies, for all $\alpha_{c_\alpha} \in Act$,

1. Whenever $P_{\sigma_p} \xrightarrow{\alpha_{c_\alpha}} P'_{\sigma'_p}$ then, $\exists\, Q'_{\sigma'_q}$, $Q_{\sigma_q} \overset{\widehat{\alpha}_{c'_\alpha}}{\Longrightarrow} Q'_{\sigma'_q}$ such that $c'_\alpha \leq c_\alpha$ and $(P'_{\sigma'_p}, Q'_{\sigma'_q}) \in \mathcal{S}$

2. Whenever $Q_{\sigma_q} \xrightarrow{\alpha_{c_\alpha}} Q'_{\sigma'_q}$ then, $\exists\, P'_{\sigma'_p}$, $P_{\sigma_p} \overset{\widehat{\alpha}_{c'_\alpha}}{\Longrightarrow} P'_{\sigma'_p}$ such that $c_\alpha \leq c'_\alpha$ and $(P'_{\sigma'_p}, Q'_{\sigma'_q}) \in \mathcal{S}$ ■

We continue with the definition of *inheritance bisimilar* processes.

Definition 4.19. P_{σ_p} and Q_{σ_q} are *inheritance bisimilar*, written $P_{\sigma_p} \preceq^s Q_{\sigma_q}$, if $(P_{\sigma_p}, Q_{\sigma_q}) \in \mathcal{S}$ for some inheritance bisimulation $\mathcal{S}$. That is

$$\preceq^s = \bigcup \{\mathcal{S} :\ \mathcal{S} \text{ is an inheritance bisimulation}\}$$

■

We have identified *protocol inheritance* and *projection inheritance* that correspond to encapsulation and abstraction. The subtle difference between the two forms of inheritance introduced so far is that under projection inheritance actions are executed without taking into account their effect, whereas under protocol inheritance they are not executed at all.

Definition 4.20. (Inheritance relations)

1. Protocol inheritance: For any processes P_{σ_p} and Q_{σ_q}, P_{σ_p} is a subprocess of Q_{σ_q} under *protocol inheritance*, iff there exists $K \subseteq Act$ such that $\mathcal{A}_3 \vdash P_{\sigma_p} \backslash K \preceq^s Q_{\sigma_q}$.

   $$P_{\sigma_p} \leq_{pt} Q_{\sigma_q}$$

2. Projection inheritance: For any processes P_{σ_p} and Q_{σ_q}, P_{σ_p} is a subprocess of Q_{σ_q} under *projection inheritance*, iff there exists $L \subseteq Act$ such that $\mathcal{A}_4 \vdash (P_{\sigma_p}[\tau_L]) \preceq^s Q_{\sigma_q}$.

   $$P_{\sigma_p} \leq_{pj} Q_{\sigma_q}$$

3. Total inheritance: For any processes P_{σ_p} and Q_{σ_q}, P_{σ_p} is a subprocess of Q_{σ_q} under *total inheritance*, iff there exists $K \subseteq Act$ such that $\mathcal{A}_3 \vdash P_{\sigma_p} \backslash K \preceq^s Q_{\sigma_q}$ and there exists $L \subseteq Act$ such that $\mathcal{A}_4 \vdash (P_{\sigma_p}[\tau_L]) \preceq^s Q_{\sigma_q}$.

$$P_{\sigma_p} \leq_t Q_{\sigma_q}$$

4. Life-cycle inheritance: For any processes P_{σ_p} and Q_{σ_q}, P_{σ_p} is a subprocess of Q_{σ_q} under *life-cycle inheritance*, iff there exist $K, L \subseteq Act$: $(K \cup \overline{K}) \cap (L \cup \overline{L}) = \emptyset$ such that $\mathcal{A}_5 \vdash (P_{\sigma_p} \backslash K)[\tau_L] \preceq^s Q_{\sigma_q}$.

$$P_{\sigma_p} \leq_{lc} Q_{\sigma_q}$$

There is given a reasonable requirement that K and L, it means set of actions that are to be encapsulated and abstracted (hidden), are disjoint. This requirement should ensure that order of projection and protocol *operators* application can be interchanged without change of the result.

When reasoning about security properties of a process and its subprocesses we find out that security requirements on subjects to perform subprocesses are at least equal to the requirements for the base process. It is implied by definition of inheritance bisimulation.

5 Results

When starting to write the thesis, one basic target was stated. We wanted to introduce a complex authorization system, or at least its basic ideas that would be suitable for strongly distributed information systems. We mean systems that are heterogeneous, consist of a number of *local* information systems with their own administration and systems that are able to process tasks that need cooperation of users and access to resources of more *local* information systems. The results achieved in the thesis can be seen in four basic areas.

There is given a general idea to solve security requirements for strongly distributed information systems. Elements of such systems (s-nodes) have their own administration, own sets of users. Resources are accessible through their own environment (operation system, DBMS). There is introduced a new global level that allows definition of tasks - workflows as mentioned above. The problem of cooperation between those two levels is solved through a conversion layer that is based on categorization of resources and subjects. All resources have a categorization that describes their content and subjects are qualified

by the same way according to their authorization. This approach is based on the fact that properties of data do not change. Categorization of data does not change until the content is changed. When we start to use new data, we add new subsets into set of categorizations, but the already existing elements do not change.

The second area involves design of Active authorization model suitable for secure workflows. The word *active* is used because the model is able to synchronize authorizations with execution of tasks. This synchronization is based on a new concept of protection states (sets of authorizations) associated with small parts of workflows - task steps. It means that there is not one protection state that exists for the whole life of the system, but appropriate protection state is activated with initialization of a task step. There are also introduced other two very important ideas:

- Strict separation of access and flow control - it starts already by different definition of privileges.
- General mechanism for transformation of authorizations between workflow and s-node.

There is introduced CCS as a formal apparatus for definition of communicating processes. This process algebra is enriched with security properties for processes and their communication. There is a chapter dedicated to approach used for incorporation of security into CCS and another one describing Secure CCS alone. Active authorization model is taken as a basis that is in an intuitive way incorporated into process algebra. We want to mention just one moment that is very strongly expressed in Secure CCS.

Assuming access control, all privileges are connected with particular processes. When we change abstraction of the model to a higher level, the privileges are hidden. It means that access control authorizations are not combined into more abstract processes. It implies that it is not necessary for a subject authorized to initiate some process to obtain access rights for s-node resources. Totally different situation comes with flow control. The privileges are combined and any subject must be authorized to work with dynamic objects - objects created during the task (workflow) processing. This distinction between access and flow control is kept very strictly.

Last contribution of the thesis is introduction of basic notions of inheritance of processes for formalism for secure processes. The idea comes from Aalst and Basten that introduced inheritance for dynamic objects in ACP algebra and also in Petri Nets. We incorporate their ideas into Secure CCS. What is it good for? We are able to easily create new workflows by using already

existing ones, while preserving abidance of security properties. It should help in administration of large systems - it is easier and more secure. The second issue is about possible dynamic changes of workflows. We are able to create subprocess for a task step of a workflow. When keeping exactly defined conditions, we are able to replace the given task step with a new one that do not have to be known id advance. Regarding security of the workflow, we just have to check properties of the subprocess.

6 Shrnutí

Při hledání tématu disertační práce, jsme se dostali až k rozsáhlým informačním systémům (workflow). Základní ideou těchto systémů je, že jsou předem definovány úlohy, při jejichž provádění mohou a musí participovat různí uživatelé. Ti jsou schopni získat přístup ke zdrojům nutným pro provedení té které části úlohy. Problém spojený s bezpečností tohoto schématu byl natolik zajímavý, že na základě určité obecné představy [18] byla vytvořena formální definice Aktivního autorizačního modelu.

Vzniklý model má určité vlastnosti, které současné modely pro kontrolu přístupu neznají. Základní myšlenkou je existence samostatného ochranného stavu pro každý krok běžící úlohy. Aby mohl být takový krok iniciován, je třeba nalézt subjekt, který splňuje určité požadavky - inicializační oprávnění. Jestliže je splní, tak získá oprávnění, která potřebuje k úspěšnému provedení tohoto kroku úlohy - uvolněná oprávnění. S každou aplikací oprávnění z této množiny se ochranný stav *zmenšuje* a s ukončením provádění kroku je zrušen a subjekt tak ztrácí všechna efektivní oprávnění pro přístup ke zdrojům systému.

Aby to nebylo tak jednoduché, tak množina uvolněných oprávnění je jasně rozdělena na dvě části. Jsou to oprávnění pro přístup k lokálním zdrojům (kontrola přístupu) a oprávnění pro přístup k dynamickým datům *vyrobených* v předchozích krocích úlohy.

Tato práce se tedy snaží identifikovat specifické požadavky na zajištění bezpečnosti v distribuovaných informačních systémech a nabízí základní bezpečnostní model, který umožňuje synchronizaci workflow úloh a *autorizačního toku*. Je navržena klasifikace základních bezpečnostní vrstev, která umožňuje snadnější uchopení bezpečnostních problémů.

Součástí práce je krátký úvod do *procesní algebry* - CCS. CCS (Calculus of Communicating Processes)[14, 10] byl navržen pro modelování komunikujících procesů. Tento formalismus je poté obohacen o navržený bezpečnostní model,

který umožňuje formální specifikaci bezpečných procesů a jejich modelování a verifikaci.

Celý text práce je rozdělen na devět kapitol podle logicky uzavřených problémových oblastí. Úvodní kapitoly předkládají určení problémů, stanovení cílů a krátký přehled bezpečnosti v informačních a databázových systémech. Další kapitola podává přehled o procesní algebře (viz. výše). Tento kalkul je použit jako základ, na do kterého je vložen formalismus pro definování bezpečnosti v distribuovaných systémech

Následující části práce již jsou autorovým příspěvkem na poli bezpečnosti v silně distribuovaných systémech. Teorie je založena na CCS a myšlenkách dědičnosti procesů. Teorie CCS a její popis je silně využit v kapitole *Secure CCS*, kde je originální kalkul modifikován tak, aby mohl být použit pro specifikaci a ověřování bezpečnostních požadavků. Další část se věnuje obecné architektuře autorizačního systému pro silně distribuované informační systémy. Tato architektura je základním kamenem následující konstrukce. Popis architektury je rozdělen na základní strukturu systému a základní komunikaci a spolupráci mezi jednotlivými vrstvami.

Další část práce představuje Aktivní autorizační model (AAM). Tento model a jeho základní ideje jsou použity jako základ pro celý autorizační systém. Původní myšlenkou bylo použít tento model tak jak je, ale ukázalo se, že je vhodné pokusit se jej vtělit také do formalismu schopného popsat komunikující procesy. Původní myšlenku lze vysledovat např. z definice vztahů mezi autorizačními kroky. Kapitola popisuje základní koncepty distribuovaných systémů, jež jsou následovány základními myšlenkami AAM modelu a jeho podrobným popisem.

Určitým mezistupněm je kapitola upřesňující způsob, jakým lze vtělit AAM do procesní algebry. Prvním předpokladem je upřesnit definici modelu AAM. Především je zavedeno jasné oddělení kontroly přístupu od kontroly toku dat a je definován způsob, jak manipulovat s bezpečnostními specifikacemi pro jednotlivé stavy a *komunikační kanály*. Současně je zaveden způsob klasifikace/kategorizace prvků systémů. Druhá část kapitoly definuje formy autorizačních informací pro stavy a definuje operátor, který umožňuje bezpečně spojovat několik stavů do jednoho - abstrakce. Na závěr jsou položeny požadavky, které musí splňovat bezpečný informační systém a v něm prováděné úlohy - bezpečnostní axiomy.

Další kapitola již zavádí bezpečný CCS. Je použit ten nejjednodušší způsob. V pořadí, ve kterém byly představovány jednotlivé pojmy CCS jsou prováděny úpravy tak, aby byly zajištěny požadavky na bezpečný informační systém. Jestliže je třeba, tak jsou nové vlastnosti dokazovány. Cílem bylo vnést do

CCS co nejmenší množství změn. Výsledný *Secure CCS* zajišťuje zachování vlastností bezpečného distribuovaného systému a zároveň zachovává jednoduchost a užitečné vlastnosti původní specifikace. Nejdůležitějšími pojmy jsou *bisimilarity*, které musely být změněny, ale původní myšlenky snad zůstaly zachovány.

Závěrečná kapitola využívá nově definovaného nástroje a spojuje jej s myšlenkami Van der Aalsta a Bastena ohledně dědičnosti procesů. Výsledkem je definice dědičnosti, která opět zajišťuje zachování vlastností bezpečného systému. Díky formálnímu nástroji je definováno několik typů dědičnosti podle způsobu porovnávání zděděných procesů s předky.

References

[1] *Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions* (Athens, GA, 1996).

[2] Bell, D., LaPadula, L.: Secure computer systems: Unified exposition and multics interpretation. Tech. Rep. MTR-2997, The Mitre Corporation, Bedford, MA, 1976.

[3] Bergstra, J., Klop, J.: Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[4] Bertino, E., Ferrari, E., Atluri, V.: A flexible model supporting the specification and enforcement of role-based authorizations in workflow management systems. In Proc. of the 2nd ACM Workshop on Role-based Access Control, 1997.

[5] Burkhard, H.: Observations on the real-world implementation of role-based access control. In 20th National Information Systems Security Conference, 1997.

[6] Demurjian, S., Ting, T., Price, M., Hu, M.-Y.: Extensible and reusable role-based object-oriented security. In Database Security, X: Status and Prospects, Chapman Hall, 1997.

[7] Denning, D.: *Secure Information Flow In Information Systems.* Phd thesis, University of Purdue, 1975.

[8] Denning, D.: A lattice model of secure information flow. *Communications of the ACM*, 236–243, 1976.

[9] Denning, D.: *Cryptography and Data Security.* Addison-Wesley, 1982.

[10] Fencott, C.: *Formal Methods for Concurrency.* International Thomson Computer Press, 1996.

[11] Ferraiolo, D., Cugini, J., Kuhn, R.: Role-based access control: Features and motivations. In Proceedings of the 11th Annual Computer Security Applications Conference (CSAC '95), 1995.

[12] Hollingsworth, D.: Workflow reference model. Tech. Rep. TC00-003, Workflow Management Coalition, Winchester, UK, 1995.

[13] Huang, W.-K.: *Incorporating Security into Workflow Management Systems.* PhD thesis, Rutgers University, CIMIC, 1998.

[14] Milner, R.: *Communication and Concurrency.* Prentice Hall, 1989.

[15] Sandhu, R.: Role hierarchies and constraints for lattice-based access controls. In Proc. Fourth European Symposium on Research in Computer Security, 1996.

[16] Sandhu, R., Feinstein, H.: A three tier architecture for role-based access control. In Proc. of the 17th NIST-NCSC National Computer Security Conference, 1994, pp. 138–149.

[17] Sandhu, R. S., et al.: Role-based access control models. *IEEE Computer*, 38–47, 1996.

[18] Sandhu, R. S., Thomas, R.: Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In Proc. of the IFIP WG 11.3 Workshop on Database Security, 1997.

7 Author's Bibliography

CVRČEK, D.: Access control in database management system. In DATASEM'98 sborník přednášek, 1998, pp. 153–163.

CVRČEK, D.: Nové přístupy k databázové bezpečnosti. In Sborník z letní školy Informační systémy a aplikace, 1998, pp. 18–35, ISBN 80-214-1205-4.

CVRČEK, D.: Problems of modeling access control in object oriented databases. In ASIS 98 Proceedings, 1998, pp. 21–26, ISBN 80-85988-27-5.

CVRČEK, D.: Access Control in Workflow Systems, In: MOSIS'99 Proceedings, MARQ Ostrava, Rožnov pod Radhoštěm, 1999, pp. 93-100, ISBN 80-85988-31-3.

CVRČEK, D.: Aktivní autorizační model, In: Sborník z letní školy Informační systémy a jejich aplikace 1999, FAST VUT Brno, Ruprechtov, 1999, pp. 46-51.

CVRČEK, D.: Active Authorization Model for Workflow System, In: Sborník prací studentů a doktorandů, roč. 5, FEI VUT Brno, Brno, 1999, pp. 73-74,

CVRČEK, D.: Active authorization as high-level control. In Proc. of the IFIP WG 11.3 Workshop on Database Security, 2000, pp. –.

CVRČEK, D.: Mandatory access control in workflow systems. In Knowledge-based Software Engineering - Proc. of the JCKBSE Conference, 2000, pp. 247–254, ISBN 1-58603-060-4.

and some other works

CVRČEK, D.: Zákon o elektronickém podpisu, In: IT System, č. 6, CCB spol.s r.o., Brno, 1999, pp. 47-51, ISSN 1212-4567.

CVRČEK, D.: Elektronický obchod, In: IT System, č. 5, CCB spol.s r.o., Brno, 1999, pp. 6-9, ISSN 1212-4567.

CVRČEK, D.: Elektronická komunikace - hrozba nebo šance?, In: IT System, č. 2, CCB spol.s r.o., Brno, 1999, pp. 15-18, ISSN 1212-4567.

8 Curriculum Vitae

Education	University of Technology Brno 1992-1997 MSc. in Computer Science and Engineering.	Brno, Czech Republic
Industrial Experience	AEC Ltd. 1998-2000 Last position: Senior software engineer, security specialist	Brno, Czech Republic
Teaching Experience	University of Technology Brno 1997-2000 Teaching assistant in J.M. Honzík course "Algorithms and Data Structures"	Brno, Czech Republic
Professional Activities	Member of IACR for year 2000 Member of expert commission for Electronic Signature Law Member of editorial board of IT System magazine	