An Authorization Model for Personal Databases

Cristian Radu Mark Vandenwauver René Govaerts
Joos Vandewalle

Katholieke Universiteit Leuven, Laboratorium ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
Tel. +32-16-220931, Fax +32-16-221855

Christian.Radu@esat.kuleuven.ac.be
Mark.Vandenwauver@esat.kuleuven.ac.be

Abstract

The multi-application paradigm in connection with a personal device is a new trend of the IC development, from which the concept of personal database has emerged. Little attention has yet been given to the conditional access to this special type of database. Therefore, we tune a well known authorization model, developed by Rabitti et al. , to the particularities of personal databases. We also present the access control mechanism that can enforce this model. The specific solution we present matches the strong restriction of a limited storage space proper to a smartcard.

1 Introduction

A multi-application personal device ($MAPD$) contains several applications that do not interfere with each other. It is the individual's representative in the framework of an electronic transaction system. The information concerning the individual is organized in a Personal Database (PDB). This is kept in the non-volatile memory of the personal device. The PDB can be considered a passive entity interacting with various active entities. These are the subscribers and the outsiders. Both organizations and individuals can subscribe to the system. During a transaction, the $MAPD$ receives an access request from an active entity, referred to as *subject*, willing to execute an *action* with regard to a piece of the PDB, referred to as *object*. An access control mechanism has to be provided with a $MAPD$ in order to check the validity of the access request against a set of authorization rules or rights. These rules specify who is entitled to do which action on what. They rely on some

previous agreed security policies in the system. Depending on the result of the evaluation, the access is either granted or denied to the subject.

In this paper we adapt the authorization model, described in [1], to the *PDB*. Our model deals only with (strong) explicit and implicit positive authorization rules. The reason is the trade-off between the number of explicit authorization rules kept within the overall system, and the simplicity of the access control mechanism. On the one hand, these rules have to be enforced, while on the other hand we have to consider the strong storage space limitations of a *MAPD*. In this respect a method is presented, which allows a self-consistent specification of the constituents of a right. This results in a efficient implementation of the procedure to process an access request. Section 2 argues the reasons of choosing a particular type of access control mechanism. Section 3 details the set of objects, subjects, actions, and their specification. In Section 4 we define the authorization rules and access requests. Section 5 formalizes the process of access request validation. Finally, Section 6 consists of possible future work.

2 A Subject View Alternative

The access control mechanism of the *MAPD* enforces a set of policies. These are related to the *PDB* in the framework of the authorization model.

As a security administration policy we choose for a centralized control. A single authorizer, trusted by everybody, called the *Issuing Authority (IA)*, controls all security aspects of the system. Among the policies for access control specification [2], we only mention the option of *least privilege policy in a closed system.* This means that in a *PDB*, everything is forbidden unless specifically allowed.

A *rule-based access control mechanism* is adopted. Its main feature is a high flexibility with respect to possible changes of security interdependencies among the right constituents. The place where the rules are located determines two major implementation approaches. The first is an object view approach, realized with an *access control list (acl)* mechanism [3]. The authorization rules are kept with the accessed objects, by the passive entity. This approach has the advantage of simplicity but requires a large storage space. The second is a subject view approach, implemented with a *subject restriction list (srl)* mechanism [4]. The authorization rules are kept with any subject in its *srl*. Each entry of the *srl* specifies an object and the corresponding action the subject may perform on it. An entry is named a *capability* or a *ticket*. The advantage is that the passive entity, in our case the "small" *MAPD*, is released of the storage space requirements. Since the ticket allows the subject to perform the specified action on the object, it has to be certified so that it can't be altered or forged. Also an adversary who intercepts a ticket shouldn't be able to use it in his own advantage. All this can be achieved by appending a digital signature from a trusted *IA*.

Because of the above we decided for the second approach. Also because the policies are not built in the mechanism, it could implement a variety of policies in a flexible way. The changing of policy is reflected only in the procedure of revoking old and issuing new

tickets by the IA to all the subjects in the system.

3 Authorization Sets

The purpose of this section is to specify the set of objects O, subjects S, and actions (or access types) A, involved in an authorization rule definition. A convenient specification of the constituents of an authorization rule is provided. We also derive the relations that can be established among the elements of the objects and actions set. Moreover, the interdependencies between the three rule dimensions are emphasized.

3.1 Object Dimension

An individual's I_k personal database, $PDB[I_k]$, can be seen as a collection of *data segments*, denoted DBS_i, $i = 1,\ldots,n$. The data segment DBS_i corresponds to the *application* A_i supported by the system. An application A_i involves a set of organizations. Each organization manages one or several *relations*, with degrees and schemes (set of attributes) according to its interests and the existing legal constraints. From each relation, only the tuple(s) concerning the individual I_k, possibly even restricted to a limited set of attributes, are copied in the $PDB[I_k]$. Therefore, the database segment DBS_i is a set of views (selected with respect to the individual I_k and projected according to a set of attributes) $R_j^{(i)}$, $j = 1,\ldots,m_i$. Each view comes from one of the databases of the organizations involved with the application A_i. We denote $T_{jk}^{(i)}[1]$, $T_{jk}^{(i)}[2]$, ... the tuple(s) concerning individual I_k in relation $R_j^{(i)}$. The above mentioned items: $PDB[I_k]$, DBS_i, $R_j^{(i)}$, $T_{jk}^{(i)}$ and the *attribute-values* of a tuple can be seen as nodes in a database granularity hierarchy, describing how granules (objects) of the personal database are organized in terms of other granules. In Figure 1 we show the personal database granularity hierarchy.

By the viewpoint of the authorization model, each node of the PDB granularity hierarchy bears two types of information: about the node itself and about the nodes at the next-lower level. Therefore, we can apply a separation process to each type of node in the database granularity hierarchy that has subordinate nodes: DBS, R, T. We obtain a hierarchy of the authorization objects types, defined as the Authorization Object Scheme (AOS) (see Figure 2). If we make a projection of the database granularity hierarchy on the AOS we obtain the Authorization Object Graph (AOG), presented in Figure 3. This represents a restriction of the Authorization Object Lattice (AOL) introduced in [1]. The set of all nodes of the AOG is the set of objects O.

There are four objects types in AOS (**setof_segments**, **setof_relations**, **setof_tuples**, and **setof_attributes**), whose objects have multiple implication links. The nodes in AOG, corresponding to these object types are shown in **boldface** and are referred to as *multi-link* nodes or M-nodes. The other object types of the AOS: Segment, Relation, Tuple and Attribute, and their corresponding objects in AOG are shown in plain in Figure 2 and 3. The nodes that have just only one implication link to the next-lower level are referred to as

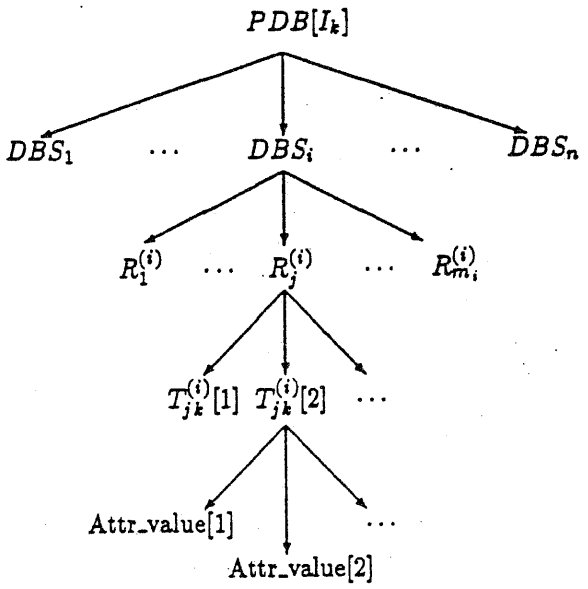

Figure 1: Personal database granularity hierarchy.

single-link nodes or S-nodes. We use the AOG to define object specifiers in a self-consistent manner.

An order relation is established over O. The notation $o_i > o_j$, where $o_i \in O$ and $o_j \in O$, means that there is an implication link from o_i to o_j in the AOG. The notation $o_i \geq o_j$, where $o_i, o_j \in O$, means that either $o_i = o_j$, or $o_i > o_j$, or there exist $o_1 \ldots, o_n$ in O, such that $o_i > o_1 > \ldots > o_n > o_j$. An implication link from o_i to o_j is represented by an directed arc from the node o_i to the node o_j in the AOG.

During an access request ar, both the requested object o and the object in the ticket o' has to be uniquely specified inside O. The *object specifier* achieves two requirements:

- To be *self-consistent*, in the sense that the answer to the question $o' \stackrel{?}{\geq} o$ can be derived by the access control mechanism of the $MAPD$ without any other information, except for the object specifiers of o and o'.
- The length of the object specifier should be acceptable in order to reduce the communication overload.

In order to specify an object o in O, one can specify the location of the corresponding node in AOG. The following information has to be provided with an object specifier:

1. The *level* L of the object o in AOG. This is an integer number between 1 and 9 (the maximum number of levels in AOG). Each mark corresponds to one floor of nodes of AOG, starting from the lowest level (the Attribute-value nodes) up to the highest level (the PDB node). The level can be encoded with four bits.

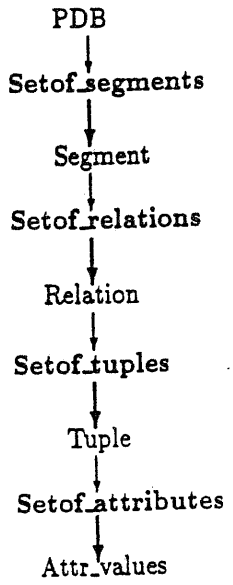

Figure 2: Authorization object scheme.

2. A *bit-flag* M/S specifying that the object o belongs to an M-node ($M/S = 1$) or an S-node ($M/S = 0$).

3. When one specifies the path to an object in AOG, one of the multiple implication links from a M-node to a S-node has to be chosen. Consequently the object specifier has to have a *numberof_selectors* field, denoted NS and maximum four *selector* fields, one for each M-node visited by the path to the object. The selectors are denoted DS, RS, TS, AS. They decide among the implication links of the objects belonging to, respectively, the **setof_segments**, **setof_relations**, **setof_tuples**, and **setof_attributes** M-nodes. Due to the strong limitations of the storage space of the PDB, we admit a limited number of 16 implication links from a M-node to a S-node. Thus each selector can be encoded as an *index* of 4 bits.

We adopt the following template as a form of an object specifier, o:

$$o = (L, M/S, NS, DS, RS, TS, AS)$$

Its dimension could range between $8 \div 24$ bits. This length can be considered acceptable from the standpoint of communication overload. Table 1 presents examples of object specifiers for different objects in AOG.

The object specifier is self-consistent. Given two object specifiers o_i and o_j, the following algorithm establishes the relation between them:

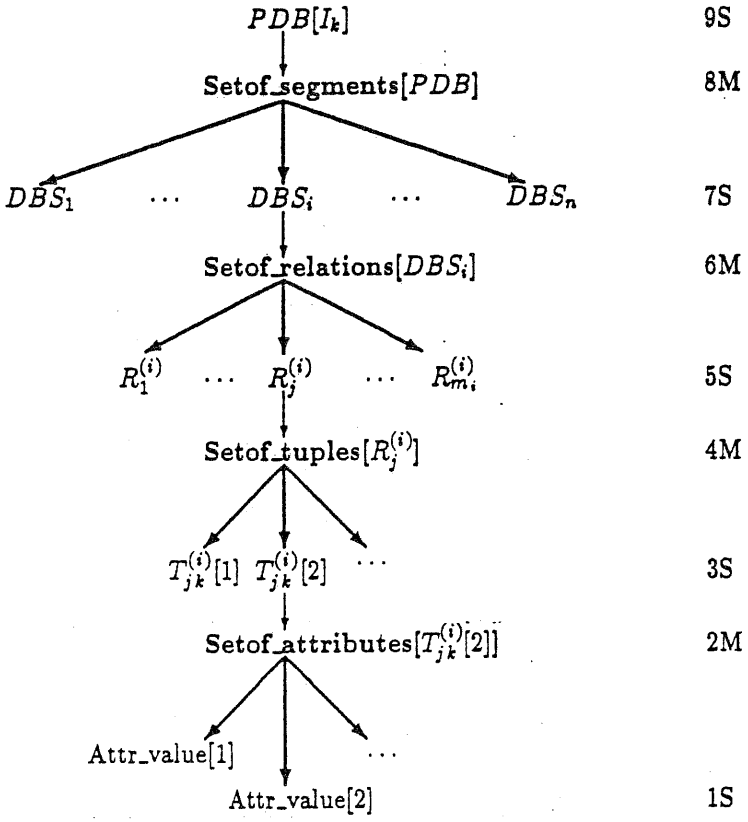

Figure 3: Authorization object graph.

Step 1 : Take the object with the highest level L. Suppose $L_i > L_j$.
Step 2 : Check if the following NS_i selector fields are equal.
$DS_i \stackrel{?}{=} DS_j, \ldots$
If the answer is YES, then $o_i > o_j$,
thus there is an implication link from o_i to o_j in AOG.

3.2 Subject Dimension

The subjects that can be distinguished with respect to the PDB are: the organizations which have subscribed to the system, the individual to whom the PDB refers (the owner of the personal device), the other organizations which did not subscribe to the system and the other individuals whether they are subscribers to the system or not.

The set of all above mentioned subjects is denoted by S. All the subjects who subscribed to the system, are identified with a unique serial number corresponding to their name, and denoted by IDS.

Object	Specifier
Setof_segments[PDB]	8M0
DBS_i	7S1i
Setof_relations[DBS_i]	6M1i
$R_j^{(i)}$	5S2ij
Setof_tuples[$R_j^{(i)}$]	4M2ij
$T_{jk}^{(i)}[2]$	3S3ij2
Setof_attributes[$T_{jk}^{(i)}[2]$]	2M3ij2
Attr_value[2]	1S4ij22

Table 1: Object specifiers.

A set of subjects having a similar behavior in the framework of a specific application supported by the system, and conversely by the personal device, is named a role, and is denoted with r. The set of all the roles in the system is denoted by $\mathcal{R}$. For each role $r \in \mathcal{R}$ we assign an unique identifier IDR.

A subject may play one or more roles. The interdependence between S and $\mathcal{R}$ is established by the Role Membership Matrix RMM:

$$RMM[s_i, r_j] = \begin{cases} True, & \text{if } s_i \text{ may act as role } r_j \\ False, & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, c_1$, $j = 1, \ldots, c_2$, where $c_1 = \#(S)$ and $c_2 = \#(\mathcal{R})$.
The set of subjects acting the role r_j is denoted by :

$$rm(r_j) = \{s_i \mid RMM[s_i, r_j] = True,\ i = 1, \ldots, c_1\}.$$

The set of roles a subject may act is denoted by:

$$sc(s_i) = \{r_j \mid RMM[s_i, r_j] = True,\ j = 1, \ldots, c_2\}.$$

A role may be involved in one or several applications supported by the system. The relation between $\mathcal{R}$ and the set of applications $A_i, i = 1, \ldots, n$ is described by the Role Relevance Matrix, RRM. An element of the matrix is defined as follows:

$$RRM[A_i, r_j] = \begin{cases} True, & \text{if role } r_j \text{ is relevant for } A_i \\ False, & \text{otherwise.} \end{cases}$$

Each application A_i is represented in the PDB by the corresponding database segment object DBS_i. Therefore, we can see the above relation as an interdependence among objects and roles. This restriction has to be considered during the tickets management operation, for all the objects $o_i \leq DBS_i$.

Due to the different behavior of roles in the framework of an application, the hierarchy of roles is also application dependent. On the one hand, this means that different identifiers have to be assigned to a role in accordance with the application in which it is involved. Thus it follows that

$$IDR[j] = \{IDRA[A_i, r_j] \mid \forall i = 1, \ldots, n \; : \; RRM[A_i, r_j] = True\}$$

This would increase and complicate the management of the certificates to a subject s_i acting the role r_j. On the other hand, there are as many Role Lattices (describing hierarchies and implication among roles [1]) as applications in the system. This would not be a problem if the storage limitations of the personal device would not be so strong. Considering the same requirement of self consistency for a role specifier (as for an object specifier), a suitable encoding to achieve a minimum in communication overload seems impossible. This is the reason for which we renounce of "ordering" roles. But, of course, this increases the number of rules kept in the system.

Considering the above mentioned remarks, we keep a unique identifier for a role IDR. Hence, the form for a subject specifier s_i can be:

$$s_i = IDS_i \; || \; \{IDR_j\}, \; \forall r_j \in sc(s_i)$$

The subject s_i proves the membership to the role r_j in $sc(s_i)$ if it has a certificate released by IA in a suitable way:

$$C_{IA}(\ldots, IDS_i \; || \; IDR_j, \ldots), \; \forall r_j \in sc(s_i).$$

3.3 Action Dimension

The actions or access types we consider in relation to a PDB are Read (R), Write (W) and Generate (G). A two-bits specifier is enough to encode any action a and $A = \{W, R, G\}$. A Generate action is used to create an object in $PDB[I_k]$. This object has the same definition as its copy concerning the individual I_k. The copy is kept with the "parent" database of the role. Hence, a Read-Definition (RD) access type is assumed for a role entitled to generate. But this action is related to the "parent" database and not to the $PDB[I_k]$.

The order relation among actions is: $G > W > R$. This means that a role able to create an object in $PDB[I_k]$ is also able to write and read it. The actions R and W are defined for all object nodes in AOG. The G action is meaningful only for the nodes of type R, T, *Attribute_value* and their corresponding **setof_** types. We specify the interdependence between objects and actions through an Action Association Matrix (AAM). $AAM[o_i, a_j]$ is $True$ if the action a_j is meaningful for the object o_j and $False$ otherwise.

4 Authorization rules

Let $\mathcal{R}$ be the set of roles, S the set of subjects, O the set of objects and A the set of possible actions.

An *explicit authorization rule* EAR is defined as a triplet $(r, o, a) \in \mathcal{R} \times O \times A$. The set of all EAR's constitutes the authorization base AB, created and managed by the IA.

$$AB = \{(r, o, a) \in S_p \subseteq \mathcal{R} \times O \times A\}.$$

An *implicit authorization rule* IAR is defined as a triplet $[r, o, a] \in \mathcal{R} \times O \times A$, where $r \in \mathcal{R}$, $o \in O$, and $a \in A$, such that:

$$\exists\, (r, o', a') \in AB : (r, o', a') \Rightarrow [r, o, a].$$

The meaning of the sign $\Rightarrow$ is that the left member implies the right member, or in other words that the $IAR = [r, o, a]$ is derived from the $EAR = (r, o', a')$.
The *scope* of an EAR (r, o', a') is the set of all $IAR's$ $[r, o, a]$ which can be derived from EAR:

$$scope(r, o', a') = \{[r, o, a] \in \mathcal{R} \times O \times A \mid (r, o', a') \Rightarrow [r, o, a]\}.$$

Three important properties have to be provided for the model:

- Completeness: this assures that the entire space $\mathcal{R} \times O \times A$ is covered by an EAR of the authorization base or by one of the scopes they are determining.

$$\mathcal{R} \times O \times A = AB \cup \left(\bigcup_{\forall\, (r,o',a') \in AB} scope(r, o', a') \right).$$

- Non-redundancy: this assures that there are no useless explicit authorization rules in the AB, that could be derived from already existing EAR.

$$\text{If } (r, o', a') \in AB \text{ then } \neg(\exists (r, o, a) \in AB : (r, o, a) \Rightarrow [r, o', a']).$$

- The interdependence relations of r, o, and a have to be respected:
  If $(\exists\ i = 1, \ldots, n\ :\ o \leq DBS_i)$ then $(RRM[o, r] = True)$ and
  $(AAM[a, o] = True)$.

In the framework of the authorization model for the PDB we accept the following implications rules:

- *Rule 1* : For any $r \in \mathcal{R}$ and $o \in O$, if $a_i > a_j$ then $(r, o, a_i) \Rightarrow [r, o, a_j]$.
- *Rule 2* : For any $r \in \mathcal{R}$ and any $a \in A$, if $o_i \geq o_j$ and the interdependence relations among r, o_i, o_j and a are respected then $(r, o_i, a) \Rightarrow [r, o_j, a]$.

A *ticket* T released by IA to any subject s acting as role r $(s \in rm(r))$ is a signed explicit authorization rule :

$$T(r, o, a) = ((r, o, a), \sigma_{IA}(r, o, a)),$$

where σ_{IA} is the secret signing transformation of the IA [5]. The set of tickets issued by IA at a specified moment of time is denoted $\mathcal{T}$ and is expressed as:

$$\mathcal{T} = \{T(r, o, a) \mid (r, o, a) \in AB\}$$

An *access request* ar of subject s to the individual's PDB is a pair consisting of :

- A ticket $T(r, o', a')$ released by IA for the subject s acting role r.
- An *authorization request* AR of subject s who wants to play role r and to perform action a on object o : $AR = \{s, r, o, a\}$.

We accept as a form for an access request, the following template:

$$ar = (T(r, o', a'), AR) = (T(r, o', a'), \{s, r, o, a\})$$

5 Basic authorization operations

The basic authorization operations are validate access requests and grant/revoke authorization rules.

We are dealing in this paper only with the first operation. It is the only one connected to the access control mechanism of the $MAPD$. The other two are more related to the IA and are presented in [6].

Validate access requests

A function v, named the *validity function*, is defined to determine if an access request ar to the PDB is valid.

$$v : \mathcal{T} \times S \times \mathcal{R} \times O \times A \rightarrow \{True, False\}.$$

Given an access request $ar = (T(r, o', a'), \{s, r, o, a\})$, if $v(ar) = True$, then subject s, acting role r is allowed to perform action a on object o. Function v is in fact an algorithm to evaluate access requests against the (explicit or implicit) authorization rules of the system. This operation is named *access request validation*. The definition of function v is :

$$v(T(r, o', a'), \{s, r, o, a\}) = \begin{cases} True, & \text{if } r \in sc(s) \wedge auth(T) = True \wedge \\ & ((r, o, a) \equiv (r, o', a') \vee [r, o, a] \in scope(r, o', a')) \\ False, & \text{otherwise.} \end{cases}$$

During the access request validation a three stage algorithm is pursued by the access control mechanism of the $MAPD$:

1. In the first stage, the certificate $C_{IA}(\ldots, IDS_i \mid\mid \{IDR_j\}, \ldots)$ of the subject s released by the IA is checked. The membership of the requested role r to the set of allowed roles for subject s is accepted, if the necessary certificate has been provided. If the subject s is allowed to act as the role r, the algorithm continues.
2. Authentication of the ticket T. This is achieved with the function $auth$, defined as follows:

$$auth : \mathcal{T} \rightarrow \{True, False\}$$

$$auth(T) = auth((r,o,a),\sigma_{IA}(r,o,a)) = \begin{cases} True, \text{ if } V_{IA}((r,o,a),\sigma_{IA}(r,o,a)) = True \\ False, \text{ otherwise,} \end{cases}$$

where $V_{IA}(message, signature)$ is the public verification transformation of the signature scheme used [5].

If the *auth*-function returns $False$, the access request is denied and the algorithm stops. Otherwise the algorithm continues.

3. The subject item s is discarded from the authorization request $AR = \{s, r, o, a\}$. The remaining three components $\{r, o, a\}$ are compared with the EAR contained in the ticket T:

$$(r,o,a) \longleftarrow \{r,o,a\}$$

$$(r,o,a) \stackrel{?}{\equiv} (r,o',a').$$

If they are equal the access is granted and the algorithm stops. Otherwise, one looks for $\{r, o, a\}$ in the *scope* of the EAR of the ticket T:

$$[r,o,a] \stackrel{?}{\in} scope(r,o',a').$$

If it does, then the access is granted, otherwise it is denied.

6 Future work

An important problem is to try to reduce the dimension of the authorization base. We see two ways how this might be achieved. The first one is to consider the (weak) negative authorization rules in the definition of the rules. A second solution might be to establish application-dependant hierarchies amongst roles. In either case, both enhancements of the authorization model have to be supported by the access control mechanism of the $MAPD$. Therefore, the key issue remains the efficient specification of the constituents of a right.

References

[1] F. Rabitti, E. Bertino, W. Kim and D. Woelk, "A Model for Authorization for Next-Generation Database Systems," *ACM Transaction on Database Systems*, Vol. 16, No. 1, March 1991, pp. 88-131.

[2] E.B. Fernandez, R.C. Summers and C. Wood, "*Database Security and Integrity*," Addison-Wesley, Reading, Mass., 1984.

[3] D.A. Curry, "Improving the Security of your UNIX System," *SRI International Tech. Report ITSTD-71-FR-90-21*, April 1990.

[4] E. Cohen, D. Jefferson, "Protection in the HYDRA Operating System," *Proc. 5th Symp. on Oper. Syst. Princ., ACM Oper. Syst. Rev.*, Vol. 9, No. 5, November 1975, pp. 141-160.

[5] C.J. Mitchell, F. Piper and P. Wild, "Digital Signatures," *in Contemporary Cryptology*, G. Simmons (Ed.), IEEE Press, 1992, pp. 325-378.

[6] C. Radu, M. Vandenwauver, R. Govaerts and J. Vandewalle, "Subject view access mechanism in the personal database", *Proc. of the 15-th symposium on Information Theory in the Benelux*, pp. 119-126, ISBN 90-71048-10-1, 1994.