AN ARCHITECTURE OF A SECURE DATABASE FOR NETWORKED COLLABORATIVE ACTIVITIES

Manabu Ando
Graduate School of Interdisciplinary Information Studies, University of Tokyo
Komaba 4-6-1, Meguro-ku, Tokyo 153-8505, JAPAN
Email: qq16206@iii.u-tokyo.ac.jp

Kanta Matsuura
Interfaculty Initiative in Information Studies, University of Tokyo
Komaba 4-6-1, Meguro-ku, Tokyo 153-8505, JAPAN
Email: kanta@iis.u-tokyo.ac.jp

Michiharu Kudo
IBM Japan Ltd., Tokyo Research Laboratory
Shimotsuruma 1623-14, Yamato-shi, Kanagawa 242-8502, JAPAN
Email: kudo@jp.ibm.com

Akira Baba
Interfaculty Initiative in Information Studies, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, JAPAN
Email: baba@hi.u-tokyo.ac.jp

Keywords: Secure Database, Collaborative Activities, Virtual Private Laboratories (VPLs).

Abstract: Open networks can be used for many purposes, for example, e-commerce, e-government, and so on. Different from those conventional applications, we consider networked collaborative activities, for example networked research activities. This application might be very useful and research activities could be significantly promoted. However, we must solve many security problems. Among those problems, we focus on an architecture of a secure database in this paper. The design of such an architecture is not a trivial task, since we want to use multiple data types and the appropriate security operations should be exexuted. In addition to this problem, some data may need multiple security properties. In this case, multiple security operations should be executed in a well-organized way (e.g. with the ensured order such as signature-then-encryption). Thus, we design an architecture of a secure database, considering multiple data types and various security operations.

1 INTRODUCTION

Recently, open networks such as Internet can be used for wider range of applications. For example, electronic commerce is a well-known usage of Internet. In this application, we can make on-line shopping or digital payment securely. Another example is an e-government. In these applications, it is a common practice to tackle security problems with the help of cryptographic protocols.

Here, we point out another important application of security technologies in the digital society: that is, secure collaborative activities over open networks. Suppose the following example; a joint research activity, in which many researchers can participate even when they are distributed over different companies, universities, or other organizations. Participants may exchange ideas via e-mail, or they may browse digital documents which describe a development specification. They may have a video conference. In addition to those basic actions, they may want to share their knowledge about the research. We can see that a lot of security problems would appear in such activities.

We can easily show some trivial security problems in such application. For example, packets in open networks should be protected with respect to integrity, confidentiality, data origin authentication, and so on. Direct use of security primitives, or VPN (Virtual Private Network), which is equipped with the primitives, can solve some of the trivial issues (Arbaugh et al., 1998). We then proceed to a non-trivial security problems in the application mentioned above.

One example of non-trivial issues is an architecture of a secure database designed for networked collaborative activities. In such activities, we usually want to exchange multiple types of data; for example text,

image, numerical values, illustrations, mathematical graphs, and so on. Each data type needs its own security property (e.g. confidentiality, integrity, non-repudiation, copyright-protection), so appropriate security operations should be executed. Text data may have to be encrypted in order to achieve confidentiality, and digital images may have to be watermarked for copyright protection. Graphs may have to be managed with their underlying numerical values in order to achieve scientific integrity. To make matters more complicated, there can be restrictions with respect to multiple security operations required for protecting a single object (e.g. not encryption-then-signature but signature-then-encryption, not watermark-then-update but update-then-watermark, etc). Thus, it is non-trivial but important to design a secure database composed of multiple data types.

However, previous works of database security did not solve, or insufficiently discussed at best, this problem. Most of the works of database security focus on access control. Access control is a significant technology in network security but multiple security properties could not be achieved by access control only. Access control by itself does not support security operations appropriate for different data types.

In order to solve the problems above, we design a new architecture of a secure database in this paper. The architecture can be used in Virtual Private Laboratories (VPLs) as a basic component of them. VPLs are the general concept of a total security system in networked research activities (Matsuura, 2002). We expect that our architecture will be used as the fundamental component in VPLs.

1.1 Related Works

In database management system (DBMS), the trigger function is used for defining a set of operations that are executed when a delete, insert, or update action is carried out. The trigger function can be also used to execute security operations, only when those actions happen. However, the trigger is not used when the other actions are required. For example, when the subject requests the read action, the trigger function could not be called, and thus the necessary operations could not be executed.

Access control is another conventional approach of database security. Fernandez et al. developed object-oriented database with access control (Fernandez et al., 1994). Jonscher and Dittrich developed database system with access control, called "Argos" (Jonscher and Dittrich, 1996). The work of Disson et al. used Role-Based Access Control (RBAC) in order to protect database (Disson et al., 2001). This work also focuses on database with access control, and then optimization or implementation issues were the main topics. The work of Tzelepi and Pangalos was the only one to protect a particular data type (Tzelepi and Pangalos, 2001). They focused on database composed of image data and RBAC to protect them. In order to implement flexible RBAC to image data sets, meta data of original image data were used.

Jajodia et al. focused on another aspect. They proposed the language for description of access control policy (Jajodia et al., 1997). Their purpose was to unite multiple structures of access control. They pointed out that there were multiple access control policies: for example, policy for file systems, policy for relational database, and so on.

In any case, all the security procedures their works care about are: (1) submission of an access request by an initiator, and (2) the response to the request by the system (grant or deny). Access control is of course an important function for database security, but access-control-only approaches are insufficient; access control by itself cannot provide multiple security operations which are required according to different data types.

1.2 Our Contribution

In this paper, we first construct an architecture of a secure database that can provide multiple security operations. We formalize the object and its attributes, and define the functions in DBMS. Those functions process the objects, and provide security operations that each data type requires. A "data type" in a conventional sense may be classified into different data types in our work; images which need confidentiality only can be recognized as a data type different from that of images which need copyright-protection as well. Next, we define the process for links among security operations because some objects may require multiple security operations. For example, one text data requires encryption for secrecy, and also requires generation of Message Authentication Codes (MAC) for integrity. In this case, two security operations should be linked in a well-organized way. Most of the existing studies ignore this linking process. Our major contribution in this paper is to provide a particular architecture for ensuring such process.

The rest of this paper is structured as follows. Section 2 describes the formalization of our architecture. In Section 3, we introduce the important linking process in our architecture with appropriate semantics. Section 4 provides a discussion on security analysis of our architecture. Finally Section 5 concludes the paper.

2 ARCHITECTURE OF DATABASE

2.1 Basic Concept of Architecture

As the problem of access control in database system, suppose the following issue: if an attacker can break access control, then the system depending solely on access control turns to be insecure. In access control system, each entity must show its authorization, and entity authentication protocols are often used. When a password system is used for entity authentication, for example, an attacker with the ability of off-line dictionary attack can be a significant threat. Once access control is broken, contents in the database are not protected at all; the attacker who has broke it can easily tamper with the contents by impersonation.

We design an architecture of secure database system which can protect or manage contents in database. As a trivial method, one may think of contents encryption. However, encryption only is insufficient. Recently, wider range of data types or contents could be used in computer and communication environment. Some data types may require encryption but others may require different security operations. Therefore, we construct an architecture of a secure database composed of multiple data types. Proposed architecture can provide multiple security operations that each data requires.

Moreover, we add an important process to our architecture. When multiple data types have been employed, we often face a scenario in which not a single but a series of security operations are required for processing a single data. For example, the data may require not only secrecy but also integrity. These properties are achieved by different security operations. However, these properties should be guaranteed in a well-organized way. Thus, multiple security operations must be linked in a well-organized manner.

2.2 Object Formalization

In our architecture, multiple data types of objects are employed. The database system should provide security operations that are appropriate to each data type. In order to achieve this, we construct the database system that can evaluate the data type of the objects. For identifying data type, we formalize the attributes of object.

The attributes are as follows:

- identification (ID) is a unique identifier of object.
- type (TP) is a data type of object, such as *text* or *image*. Moreover, we deal with the data generated by security operations as an object. For example, MAC is also the object, and TP of this object is described as *MAC*. Let m be the number of types our system provides.
- register flag (RF) is a flag which indicates a registration status: "+," or "−," or otherwise "null". "+" means that this type of object can be processed in the system. On the other hand, "−" means that the system cannot process this data type. Finally, null is the initial value of this attribute.
- state (STO) is a state of object. This value has three types.

  Completed This value indicates that all the required operations are completed.

  Insufficient This value indicates that some of the required operations are executed but that others are not.

  Not_Executed This value indicates that none of the required operations have been executed.

By using these attributes, we define an object as follows:

Definition 2.1. Object (OBJECT) consists of 4-tuple, (ID, TP, RF, STO). This means that OBJECT with a unique identifier ID is an object of type TP, and the state of OBJECT is STO. If RF is "+", this object can be processed, whereas if "-", this object cannot be processed in the system.

Example 2.1. Suppose the object, OBJECT=(0001, text, +, Not_Executed). This object is a *text* data, which can be processed in the system, and security operations have not been executed. On the other hand, the object described as OBJECT=(0002, image, +, Completed) is an *image* data, and security operations have been completely executed.

2.3 Subject Formalization

In this paper, we assume that subjects belong to groups. Each subject may belong to more than one group. Moreover, the groups may form a hierarchy. Figure 1 illustrates an example of such hierarchy.

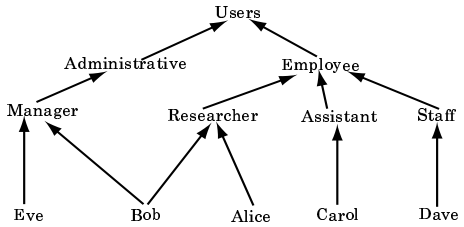

Figure 1: An Example of Group Hierarchy

The authorization is assigned to group, not to each subject. Each subject is controlled by assigned authorization. The database system has the function that takes subject information as an input and returns

the relationship between the subject and group. This function refers group hierarchy as is illustrated in Figure 1.

2.4 Class Structure

We define three types of classes: **Control Class** (CC), **Data Class** (DC), and **Operation Class** (OC). The outline of class structure is illustrated in Figure 2. We define m DC's and n OC's, where $n > m$.

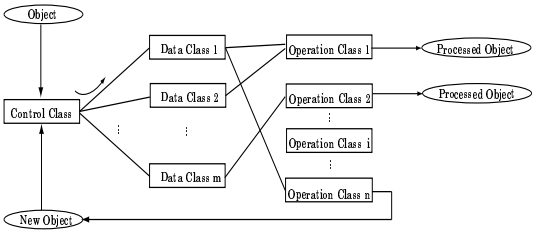

Figure 2: Outline of Class Structure: For 1 CC, m DC's and n OC's are defined. DC is defined for each data type. If necessary, CC, DC, and OC are called recursively.

2.4.1 Control Class

CC works mainly for identification of data type or state of object. In addition to the identification, CC decides necessary DC (See Figure 3). This class has four functions:

- Ident_TP: This function takes ID and TP as inputs, and identifies data type. After evaluating inputs, it returns a 2-tuple, (ID,TP), as Object_Type.
- Ident_STO: This function takes ID and STO as inputs. After evaluating inputs, it returns a 2-tuple, (ID,STO), as Object_State.
- Change_RF: This function takes TP as an input, and evaluates it. It returns RF: "+" or "−". Moreover, it changes the RF value of the object. If the evaluation result is "−", then this function returns "error".
- Instruct_DC: This function takes Object_Type as an input, and returns necessary DC's.

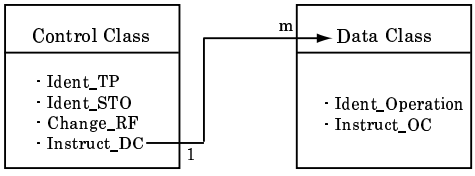

Figure 3: Control Class: CC identifies data type or state of the object. Moreover, using the list of registered data type, CC changes the value of RF. From the attributes of object and the outputs of other functions, Instruct_DC function decides necessary DC's.

2.4.2 Data Class

DC mainly evaluates the state of objects, and decides necessary OC's (See Figure 4). This class has two functions:

- Ident_Operation: This function takes Object_Type and Object_State as inputs. Evaluation is processed as follows:

  Case 1. If STO = Insufficient ∪ Not_Executed, then this function returns Operations_Need.

  Case 2. If STO = Completed, then Ident_Operation returns Not_Need. The system finishes the process when Ident_Operation returns this value.
- Instruct_Operation: This function takes Object_Type, Object_State, and Operations_Need as inputs. It evaluates the inputs, and decides necessary OC's.

Different DC's are defined for different data types. For example, the DC for *text* and the DC for *image* are defined independently.

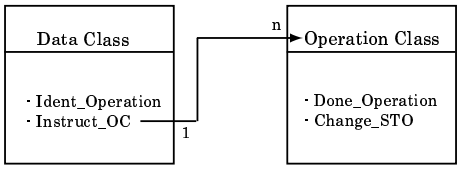

Figure 4: Data Class: DC is defined for each data type. Instruct_DC function in CC calls DC. DC evaluates the state of object, and Instruct_OC decides necessary OC's.

2.4.3 Operation Class

OC mainly calls necessary functions or modules for security operations (See Figure 5). For example, when encryption is required, encryption module, such as Java Cryptography Extension, is called. In addition to this, OC changes STO value. This class has two functions:

- Done_Operation: This function takes the object as an input and it calls functions or modules for executing security operations. It returns processed objects and Done_Flag which means all operations are completely executed.
- Change_STO: This function takes Done_Flag as an input, and it changes the value STO as Completed.

3 LINK AMONG OPERATIONS

In our architecture, a series of security operations should be executed properly in order to achieve mul-

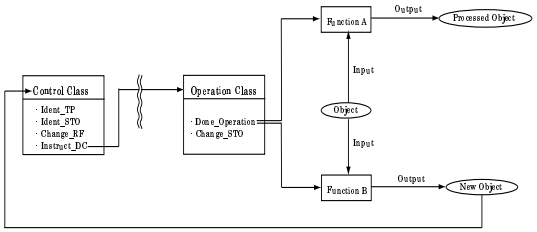

Figure 5: Operation Class: OC calls necessary modules for executing security operations, and changes STO value in the object. When some operations generate a new object, CC and DC are recursively called.

tiple security properties. Simple example is described as follows:

Example 3.1. Suppose TP of an object is *text* and the object requires secrecy and integrity. In this case, encryption and generation of MAC are the appropriate security operations. Thus, the database system should provide these security operations for the object.

However, it is insufficient even if each property is achieved independently. For security reasons, multiple different operations, which satisfy different properties, should be linked in a well-organized way. This tip is well known for secure communication (Bellovin, 1996), (Kent and Atkinson, 1998), (Madson and Glenn, 1998a), and (Madson and Glenn, 1998b) but most of the studies in database security ignore this process. In order to construct link process in our architecture, we use the access control method, called Provision-Based Access Control (PBAC). Note that we do not use PBAC itself, but use the concept and functions of PBAC. We refer to our proposed process as "**Provision-Based Linking Process**."

3.1 Provision-Based Access Control

In most access control systems, authorization is specified using binary decisions, "yes" or "no." Such systems assume the following model: "A user makes an access request of a system, and the system either authorizes the access request, or denies it."

However, today's rapid expansion of Internet or network makes traditional access control models cumbersome and inflexible (Ghosh, 2000). In this situation, some other operations are also required, in addition to binary access decisions. In order to realize such requirements, Kudo and Hada proposed Provision-Based Access Control (PBAC) model, which expands traditional access control model and adds necessary actions as provisional actions (Kudo and Hada, 2001). Using PBAC, flexible or fine-grained access control and policy specifications are realized. Next, we introduce the summary of PBAC model.

3.1.1 Primitives and Functions

Table 1 shows component primitives in PBAC. Functions in PBAC are described as follows.

- eval_pvn: eval_pvn takes an access request as an input. After evaluating access request using access control rules, eval_pvn returns provisional actions specified in the selected rules.
- eval_prm: eval_prm works similar to eval_pvn, except that eval_prm returns the permission flag specified in the selected rules.
- provisions:

$$\mathsf{state}\Big(\{\bigcup_{\{\mathsf{eval\_pvn}(q)\}} \mathsf{provision}(\mathsf{eval\_pvn}(q))\}\Big)$$

  provisions takes an access request q as an input and returns SEQ. state function generates a SEQ from a set of operations, and assigns the current state s_c to the first CMD of SEQ. $\bigcup$ represents the union of arguments.
- provision:

$$\begin{cases} \phi : if\ q = \phi \\ \mathsf{err}(q) : if\ q \neq \phi\ \wedge \\ \quad \mathsf{prov\_prm}(q) = \mathsf{deny} \\ \{\bigcup_{\{\mathsf{eval\_pvn}(q)\}} \mathsf{provision}(\mathsf{eval\_pvn}(q)), q\} : \\ \quad if\ q \neq \phi \wedge \mathsf{prov\_prm}(q) = \mathsf{grant} \end{cases}$$

  provision is inductively defined. The first formula means that if a command is empty, then return the empty SEQ. The second formula means that if the command is not empty and prov_prm function returns deny, then return err showing that an error occurred in evaluating provisional actions. The third formula is a recursive definition that means provision calls itself recursively with the results of eval_pvn(q) as the argument.
- prov_prm: prov_prm takes a command as an input, and returns grant if the permission flags of all provisional actions are grant, or returns deny if any permission flag of the provisional actions is deny.

Based on these primitives, the following two components are defined.

Definition 3.1. Access Control Policy Information (ACPI) is a 7-tuple, (OBJ, SBJ, ACT, PRM, OPS, CXT, SBJ2). This means that if PRM is grant, then SBJ can perform ACT on OBJ under CXT, provided that all of OPS are executed. If PRM is deny, then SBJ cannot perform ACT on OBJ under CXT, but all of OPS must still be executed. SBJ2 refers to the permission grantor.

Definition 3.2. Access Request (AR) is a 4-tuple, (OBJ, SBJ, ACT, CXT). AR indicates whether SBJ can perform ACT on OBJ under CXT or not.

ST	A symbol that represents the state of access control system. This symbol changes whenever any of the data in the system is changed.
SBJ	An initiator information, such as user ID.
OBJ	A resource identifier, such as file name.
ACT	An access mode permitted on the target object, such as *write*, *read*.
PRM	A flag indicating whether the access can be granted or denied. PRM={grant, deny}
CXT	Context data, such as time.
OP	(OBJ,SBJ,ACT,CXT), which means that SBJ performs ACT on OBJ under CXT
OPS	A set of OP's.
CMD	$(ST_i, OBJ, SBJ, ACT, CXT, ST_{i+1})$, which means SBJ performs ACT on OBJ under CXT in state ST_i and state is changed to ST_{i+1}.
SEQ	An ordered list of CMD's.

Table 1: Component Primitives in PBAC

3.1.2 Semantics of PBAC Model

In this section, we introduce the semantics of PBAC model with a concrete example. Suppose the following example:

Example 3.2. The user, *Alice*, belongs to the group of *Researcher*, and Alice submits Access Request R1.
R1(*document1*, *Alice*, write, -)
Suppose there are two access control rules P1, P2 in ACPI.
P1(*log*, *Researcher*, create, grant, -, -, -)
P2(*document1*, *Researcher*, write, grant,
(*log*, *Researcher*, create, -), -, -)

When the system receives R1, it evaluates R1, using eval_prm and provisions functions. First, eval_prm(R1) returns grant, since eval_prm function selects P2 as an appropriate rule for R1. Second, provisions(R1) function calls provision function with the argument (*log*, *Researcher*, create, -) as a new access request. In this case, the third formula of the provision function matches this argument. Thus, provision function is called recursively, and P1 is selected as an appropriate rule. Finally, provisions function returns permission grant and a command sequence SEQ.

3.2 Provision-Based Linking Process

In order to implement Provision-Based Linking Process, we can use functions in PBAC. Suppose the following examples:

Example 3.3. The user, *Alice*, belongs to the group *Researcher*, and *Alice* submits Access Request, R1, to database.
R1(*document1*, *Alice*, write, -)
For this Access Request, the system administrator forces *Alice* to execute additional security operations; encryption of *document1* and generation of MAC of *document1*. In this case, the specifications of ACPI are:
P1(*document1*, *Researcher*, write, grant,
((*document1*, *Researcher*, encrypt, -),
(*MAC*, *Researcher*, generate, -)),-, -)
P2(*document1*, *Researcher*, encrypt, grant, -, -, -)
P3(*MAC*, *Researcher*, generate, grant, -, -, -)

These rules show that *Alice* has write permission on *document1*, provided *Alice* executes encrypt action of *document1* and generate action of *MAC*. This process is similar to that in PBAC, so we can use functions in PBAC to construct our Provision-Based Linking Process. In the following sections, we define Provision-Based Linking Process.

Note that we define the database architecture and its functions in Section 2. These functions and PBAC functions must interact in order to implement flexible process.

Provision-Based Linking Process mainly works on Instruct_OC in DC, Done_Operation and Change_STO in OC.

3.2.1 Instruct_OC Function

Instruct_OC function decides necessary OC's which can execute required operations. During this process, Instruct_OC function refers to rules which define the required operations. As these rules, we can use the specification of ACPI in PBAC. ACPI is described as (OBJ, SBJ, ACT, PRM, OPS, CXT, SBJ2) (see Definition 3.1).

Instruct_OC function takes Object_Type, Object_State, and Operations_Need as inputs, and refers to (OBJ,ACT,OPS) in ACPI. From the data type, state, and required operations specified in OPS, Instruct_OC can decide an appropriate OC which executes proper security operations.

3.2.2 Done_Operation Function

Done_Operation function executes security opera-

tions, or calls necessary modules. Required security operations can be multiple, and Done_Operation function should execute a series of security operations in a well-organized way.

In this process, we use ACPI, eval_prm, provisions, and provision in PBAC. These functions process in accordance with the model of PBAC. Done_Operation function calls functions in PBAC if necessary and receives output values.

Detail process is as follows:

1. Done_Operation takes the object as input (see Section 2). It creates Access Request, using the object and required access mode.
2. Done_Operation calls eval_prm, and it returns permission flag using ACPI.
3. provisions is called, and it processes Access Request (provision function is also called).
4. provisions returns permission flag and required operations as provisional action.
5. Receiving the output of provisions, Done_Operation calls necessary modules. If the permission flag is deny, Done_Operation stops process.

Done_Operation function also returns Done_Flag which means all operations have been executed (see Section 2). However, if some operations generate new objects, then all of the operations are not completely executed. In such a case, Done_Operation function returns Insufficient_Flag which means that all the operations have not been completely executed. Change_State function takes this flag as input and changes the value of STO into Insufficient, and the process to original object is stopped.

3.2.3 Change_STO Function

Change_STO function evaluates the output of Done_Operation function. However, if an operation generates another object, this object may also require another operation. In this case, CC is called recursively, and new object must be processed in DBMS. STO of new object may be Not_Executed. In such a situation, the original object has not been completely processed, since all of the required operations are not executed. Change_STO function should stop the process to the original object.

In PBAC, the following policy is defined: the access is not granted, provided all required operations are completely executed. In Change_STO, we define the following policy: STO value of the original object is not changed, provided STO values of all derived objects turn into Completed.

In order to implement this policy, Done_Operation function returns Done_Flag or Insufficient_Flag. If Change_STO function takes Insufficient_Flag, it changes STO value of the original object into Insufficient and stops process to the original object. After processing all derived objects, Done_Operation function returns Done_Flag. If Change_STO function takes this flag, it changes STO value of the original object into Completed.

4 DISCUSSION

4.1 Attacker

In this paper, breaking access control does not imply a complete tampering with the system. Breaking access control implies: (1) the attacker can break access control which works on database access, and (2) the attacker can impersonate to the authorized subject, but (3) the attacker cannot always compromise whole the system. When the attacker can break the access control, he can control the contents in the database but he does not compromise all of the system. We refer to this attack as breaking access control.

Suppose the traditional access control for database system. If this system is attacked by breaking access control, the contents in the database are easily controlled, since the contents are not protected at all. The attacker could not control whole the system but he can control the contents.

Our architecture can provide security for the contents in the database, once the system is attacked by breaking access control. In order to demonstrate this feature, we discuss security of our architecture in this section.

4.2 Security Analysis

In Done_Operation function, we use functions in PBAC (eval_prm, provisions, and so on) in order to construct Provision-Based Linking Process. In these functions, provisions function returns permission flag, grant or deny. Moreover, if any permission flag of the provisional actions is deny, then Access Request is also denied (See Section 3).

Even if the attacker can break access control and impersonate to the authorized subject, he cannot tamper with all of the contents in database. Provision-Based Linking Process can protect the contents in the database from the attacker. The attacker only can tamper with the contents which impersonated subject has the authorization to.

Moreover, the attacker cannot tamper with the contents which the impersonated subject is authorized to. Suppose that the encryption key used by Provision-Based Linking Process is stored in an IC card, and that the authorized subject must use this key in order to process an object. The assignment of such a rule

could be used for the protection of the contents. If the attacker can break access control, but cannot get the encryption key stored in an IC card, then the attacker cannot tamper with the contents which the subject has the authorization for.

Therefore, Provision-Based Linking Process can provide security for the contents, even if the attacker can break access control. This security property cannot be achieved in most of the previous works in database security.

5 CONCLUDING REMARKS

In order to promote collaborative activities over open networks, we introduced an architecture of a secure database. In such activities, multiple types of data may be used, and each data type requires its own security properties achieved by a series of proper security operations. However, previous works in database security cannot achieve this. In our architecture, wider range of data types in the context of information security can be employed and necessary security operations can be executed in a well-organized way.

Moreover, we introduced the notion of Provision-Based Linking Process, which can realize links among multiple security operations. This process is based on Provision-Based Access Control but it is not an access control technology. The main contribution is well-organized linking processes among security operations; for example, if an update process of a text file needs subsequent generation of a digital signature by a particular subject, this signature generation is ensured by the mechanism of Provision-Based Linking Process. Thus a wide range of security properties can be achieved. Furthermore, the system administrator can assign the rules or policies about operations flexibly, so fine-grained requirements can be achieved. The proposed architecture can be employed in Virtual Private Laboratories (VPLs) as a basic component.

REFERENCES

Arbaugh, W. A. et al. (1998). Security for Virtual Private Intranets. *IEEE Computer*, 31(9):48–55.

Bellovin, S. M. (1996). Problem Areas for the IP Security Protocols. In *the Sixth USENIX Security Symposium*.

Disson, E. et al. (2001). A Role-Based Model for Access Control in Database Federations. In *Information and Communications Security–ICICS2001*, pages 429–440. LNCS2229.

Fernandez, E. B. et al. (1994). A Model for Evaluation and Administration of Security in Object-Oriented Databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):275–292.

Ghosh, A., editor (2000). *E-Commerce Security and Privacy*, chapter 8. Kluwer Academic Publishers, Boston.

Jajodia, S. et al. (1997). A Unified Framework for Enforcing Multiple Access Control Policies. In *International Conference on Management of Data*, pages 474–486.

Jonscher, D. and Dittrich, K. R. (1996). Argos–A Configurable Access Control System for Interoperable Environments. In *Ninth Annual IFIP TC11 Working Conference on Database Security*, pages 43–60.

Kent, S. and Atkinson, R. (1998). *IP Encapsulating Security Payload(ESP)*. RFC2406.

Kudo, M. and Hada, S. (2001). Access Control Model with Provisional Actions. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A:295–302.

Madson, C. and Glenn, R. (1998a). *The Use of HMAC-MD5-96 within ESP and AH*. RFC2403.

Madson, C. and Glenn, R. (1998b). *The Use of HMAC-SHA-1-96 within ESP and AH*. RFC2404.

Matsuura, K. (2002). Virtual Private Laboratories: Concept and Two Building Blocks. In *the 2002 IEEE International Engineering Management Conference (IEMC-2002)*.

Tzelepi, S. and Pangalos, G. (2001). A Flexible Role-Based Access Control Model for Multimedia Medical Image Database Systems. In *Information Security–ISC2000*, pages 335–346. LNCS2200.