Information Flow Controls vs Inference Controls: An Integrated Approach

F. Cuppens[1] and G. Trouessin[2]

[1] ONERA-CERT, 2 Av. E. Belin, 31055 Toulouse Cedex, France,
email: cuppens@tls-cs.cert.fr
[2] CESSI CNAM-TS, 14 Place Saint Etienne, 31000 Toulouse, France

Abstract. This paper proposes a formal method for modeling database security based on a logical interpretation of two problems: the (internal) information flow controls and the (external) information inference controls. Examples are developed that illustrate the inability of "classical" security models such as non-interference and non-deducibility to completely take into account the inference problem, because both are too constraining: the former model leads to the existence problem, whereas the latter one leads to the elimination problem. The causality model, which has been developed to solve the information flow control problem by considering that "what is known, must be permitted to be known", does not also explicitly take into account the inference problem. But we show that it is possible to extend causality so that inference can in fact be solved by formalizing the security policy consistency in the following way "any information must not be both permitted and forbidden, to be known". However, some difficulties remain if we do not consider that a subject can perform not only valid derivations but also plausible derivations. In particular, we show that classical solutions to the inference problem such as use of polyinstantiated databases are not plainly satisfactory, unless the security policy is able to estimate how it is plausible that an abductive reasoning can occur.
Keywords: Security model, Information flow control, Database security, Inference control, Modal logic.

Introduction

An application that has been of particular interest since the beginning of work on secure computer systems is the implementation of a secure database management system (DBMS). To design and construct a secure DBMS, we need a formal model in order to define the security requirements, to have a precise description of the behavior desired of the security relevant portions of the DBMS and to have a means to prove that these portions of the DBMS are secure with respect to the security requirements.

The initial works of Hinke and Schaefer [26] and Grohn [22] provide an interpretation of the Bell and LaPadula model [2] for a relational DBMS. These first applications of a security model to a DBMS are still restrictive because the Bell

and LaPadula model was not designed to deal with several important problems, among them we state:

1. Information can be passed by subtle and indirect means which the Bell and LaPadula model cannot detect.
2. Users can derive secret information from that to which they have legal access.

Afterwards, non-interference [21] and non-deducibility [34] models have been developed. They present formal frameworks which try to solve these problems. Concurrently, several realizations of secure databases were initiated. These projects generally enclose a formal verification of the database operations against the security properties of the policy model. This is, in particular, the case of the Seaview [14, 15] and LDV [24] projects.

The Seaview verification effort is described in [37]. The Seaview specifications contain a formal policy model of the security requirements for multilevel secure databases as well as an abstract description of the database operations. Seaview does not use a classical model of information flow control such as non-interference or non-deducibility but rather an ad-hoc model. In this model, to infer that the global system is secure, it must be proved that the initial state is secure and that each command is secure. To prove that a command is secure, it must be proved that it satisfies the *secure-states* and *secure-transitions* predicates. To prove the former predicate, fourteen properties must be satisfied and to prove the latter, sixteen properties must be satisfied. The problem with the formal security policy model used by Seaview is the absence of a general definition of the security constraints such as that proposed in non-interference and non-deducibility models.

On the other hand, the LDV project is controlled by the basic LOCK policy which satisfies the requirements of the non-interference formal model. This approach provides good assurance that the design is secure. However, we consider that classical models of information flow (such as non-interfence and non-deducibility) are too constraining to realistically take into account the security problems in a DBMS. Section 1 states through examples this point of view. In section 2, we show that in order to have a correct model of the security requirements in a DBMS, it is more convenient to split up the problem of confidentiality into two sub-problems:

1. Internal information flow controls.
2. Inference control.

This decomposition was already suggested by Denning in [13]. In fact, our previous analysis in [9] shows that non-interfence and non-deducibility models try to jointly solve these two sub-problems, but we consider that they do not provide a satisfactory solution to any of them.

In sections 3 and 4, we give a logical interpretation of these two problems. To analyze the confidentiality of a system in a logical context, we need a formal definition of three concepts:

- The knowledge of each subject, we denote it K_A.

- The permission to know of each subject, we denote it PK_A.
- The prohibition to know of each subject, we denote it FK_A.

In the context of the logic of security, confidentiality is defined by a logical formula $K_A\varphi \rightarrow PK_A\varphi$ that could be read:

If A knows φ then A should be permitted to know that φ

In [4], we provide a semantics for this logical formula which leads to a new security condition called causality. In section 3, we show that the security enforced via causality provides a satisfactory solution to the problem of **internal** information flow control but it does not deal with the inference control problem. Therefore, the aim of section 4 is to show how to extend causality in order to take into account this problem. We show that in the context of the logic of security, inference control is defined by the formula $\neg(PK_A\varphi \wedge FK_A\varphi)$ that could be read:

A cannot both have the permission to know φ and the prohibition to know φ

The enforcement of this condition guarantees that no inference channel exists which uses valid derivation. Thus, by combining the causality and consistency requirements, we obtain a general and complete formal method for modeling database security. However, some difficulties remain because a subject can also perform plausible derivations. In particular, we analyze some potential solutions such as the incompleteness and/or the polyinstantiation of the database. We can show that with the help of abductive reasoning some information can be illegally deduced, as stated in [18], unless some supplementary measures have been taken, such as imprecise assessment of potential abductive information. Finally, section 5 concludes on further work that remains to be done.

1 Drawbacks of "classical" information flow models

It is generally considered (see [23] for instance) that computer security is concerned with the transmission of information through a computer system. Goguen and Meseguer with non-interference [21] and, following them, Sutherland with non-deducibility [34] presented frameworks for identifying general flows of information through a computer system, and suggested policies that would disallow some of them.

Actually, the main difference between non-interfence and non-deducibility is that these definitions do not agree on information flows that must be disallowed. However, we can bring out several common points between these two definitions:

1. At the initial time, each subject A perfectly knows all the possible behaviors (traces) of the system. Then, A observes the system by performing inputs and receiving outputs. Thus, for each trace t of the system, A has a partial view of this trace that we called the restriction of t to A and we write it $t\lceil A$. Finally, we can define what A can infer from its observation in trace t by defining its **knowledge** in t:

A subject A knows a piece of information φ in a given trace t if and only if φ is true in all traces t' such that $t\lceil A = t'\lceil A$ (that is to say t and t' are indistinguishable according to A's observation).

2. For each subject A, both non-interference and non-deducibility aim to protect a given set of secret information (see Figure 1). According to these two definitions, this set is the sequence of inputs performed by another user B. In [3], we showed that, for these two definitions, the protection of this set of secret information means the enforcement of an **ignorance** condition for A, that is to say:
   - **non-interference:** B does not interfere with A if and only if A **does not know** that B has performed any input in the system.
   - **non-deducibility:** A does not deduce anything on B if and only if for every possible behavior[3] b_i of B, A **does not know** whether B had a behavior different from b_i.

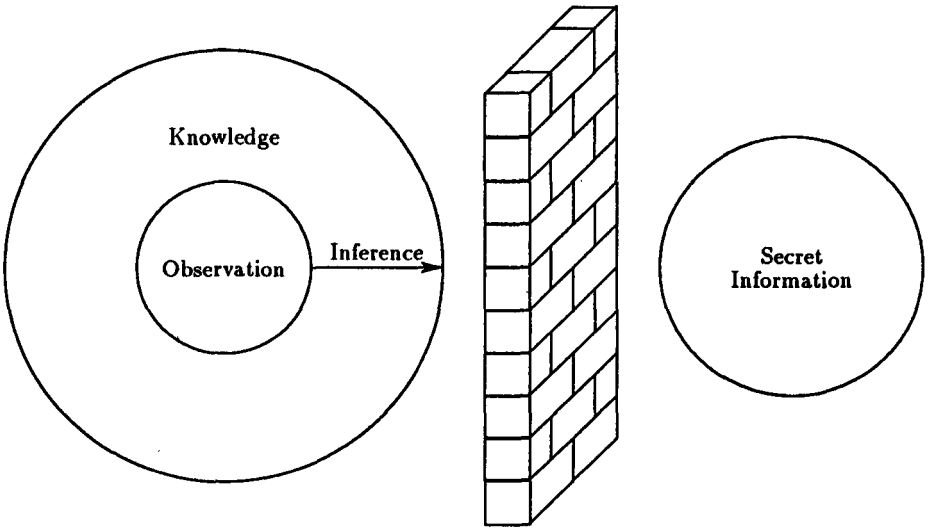

Fig. 1. Classical representation of confidentiality

In [4], we showed that there exist several problems with non-interference and non-deducibility when we use these definitions to control information flow through a computer system. In particular, we showed that these definitions disallow any kind of dependency between unclassified information and secret infor-

[3] b_i is a possible behavior of B if and only if there exists a trace t such that the sequence of inputs performed by B in trace t is equal to b_i, i.e. $t\lceil B_i = b_i$

mation. Moreover, these definitions require implicit assumptions on the subjects' behavior:

- For non-interference, it is always possible that B does not perform any input.
- For non-deducibility, inputs performed by A and B are always compatible.

In this paper, we want to show that, due to these problems, these two definitions are not adapted to model database security.

1.1 Example 1: The existence problem with non-interference

Let us consider, in this first example issued from [28], an ordinary database relation, *Mission*, with three attributes, *Starship*, *Objective* and *Destination*, with *Starship* being the key: this means that for each starship there is at most one tuple in the *Mission* relation giving us the *Starship*'s unique *Objective* and unique *Destination*. For example, the tuple $\langle Intergalactic, Exploration, Talos \rangle$ denotes that the starship *Intergalactic* has set out for an *Exploration* of *Talos*. This entire tuple gives us the mission of *Intergalactic*, as shown in Table 1.

Starship	Objective	Destination
Enterprise	Spying	Rigel
Intergalactic	Exploration	Talos

Table 1. The ordinary *Mission* relation

Let us now consider a multilevel relation which attempts to represent the same information as in the ordinary *Mission* relation, but in a context in which all the facts recorded in the database, denoted *DB*, are classified according to their confidentiality level. Suppose that there are only two classification levels: the high level or *Secret* level (denoted *S*) and the low level or *Unclassified* level (denoted *U*). Following the example issued from [28], each attribute value of each tuple can be associated with a given confidentiality level so that the previous *Mission* relation becomes a new multilevel *SOD* relation (as shown in Table 2). Each tuple of the *SOD* relation can also be associated with a classification, the *Tuple Classification* (or *TC*), which is the highest classification level of the classification levels of all the attribute values of the tuple (as indicated in Table 2).

Then, it is possible to decompose a multilevel relation in a set of single-level relations [28, 11]. In the case of the example shown in Table 2, it can thus be considered that two distinct databases (DB_S and DB_U) are managed by the secure DBMS to represent the original database *DB*:

- DB_S contains the *Secret* data of *DB* to which only any *Secret* user, $users_S$, has access;

Starship	Objective	Destination	TC
Enterprise U	Spying S	Rigel S	S
Intergalactic U	Exploration U	Talos U	U

Table 2. The multilevel *SOD* relation

- DB_U contains the *Unclassified* data of DB to which both any $user_S$ and any *Unclassified* user, $user_U$, have access;

Each of these two databases is able to give its own answer to the following request:

Request1: SELECT $\langle Starship, Objective, Destination\rangle$ FROM SOD
Answer1.S: $\langle Enterprise, Spying, Rigel\rangle$ (if $user_S$ has sent *Request1* to DB_S)
$\langle Intergalactic, Exploration, Talos\rangle$
Answer1.U: $\langle Intergalactic, Exploration, Talos\rangle$
(if $user_U$ has sent *Request1* to DB_U)

and each database can also give its own answer to the following request:

Request2: SELECT $\langle Starship\rangle$ FROM SOD
Answer2.S: $\langle Enterprise\rangle$ (if $user_S$ has sent *Request2* to DB_S)
$\langle Intergalactic\rangle$
Answer2.U: $\langle Enterprise\rangle$ (if $user_U$ has sent *Request2* to DB_U)
$\langle Intergalactic\rangle$

Let us assume that DB is complete with respect to SOD (as it is the case in Table 2). This means that if a given *Starship* is stored in DB then its *Objective* and *Destination* are also stored in DB. Since $DB = DB_S \cup DB_U$, *"DB is globally complete with respect to SOD"* means that SOD is represented in, either DB_S, or DB_U, or both. In other words:

$$[\forall star, (Starship(star) \in DB) \Rightarrow (\exists obj, \exists dest, SOD(star, obj, dest) \in DB)]$$

or equivalently:

$$[\forall star, (Starship(star) \in DB) \Rightarrow (\exists obj, \exists dest, SOD(star, obj, dest) \in DB_S \vee SOD(star, obj, dest) \in DB_U)]$$

Suppose now that $user_U$ sends the following request to DB_U:

Request3: SELECT $\langle Destination\rangle$ FROM SOD
WHERE $Starship = Enterprise$
Answer3a.U: Unknown (if $user_U$ has sent *Request3* to DB_U)

Although such an answer does not seem to provide any information, $user_U$ can use the hypothesis that DB is complete to build the following reasoning.

From:

$K_{user_U}[\forall star,(Starship(star) \in DB)$
$\Rightarrow (\exists obj, \exists dest, SOD(star, obj, dest) \in DB)]$

and by considering the information given by *Answer2.U*:

$$K_{user_U}[Starship(Enterprise) \in DB]$$

$user_U$ can derive that:

$$K_{user_U}[\exists obj, \exists dest, SOD(Enterprise, obj, dest) \in DB]$$

and from *Answer3a.U*:

$$K_{user_U}[\forall dest, Destination(Enterprise, dest) \notin DB_U]$$

Finally, by using the fact that $DB = DB_U \cup DB_S$, $user_U$ can derive that:

$$K_{user_U}[\exists dest, Destination(Enterprise, dest) \in DB_S]$$

If the hypothesis is made that *Secret* tuples are only introduced in DB by the way of *Secret* inputs[4], this kind of reasoning subsumes that another user, $user_S$ (whose clearance is *Secret*), has performed some secret input in DB (in DB_S to be more precise). In [9], we obtain a similar result in developing a different argumentation based on the *restricted* value first introduced by Sandhu and Jajodia in [31]. Now, let us consider that the non-interference model is used, this means that:

- $user_U$ must not know that *Enterprise*'s destination is *Rigel*, because the classification level of the destination of *Enterprise* is secret;
- $user_U$ must not even know that the classification level of *Enterprise*'s destination is secret, because this would imply that a secret input has been performed. In our example, from the non-interference point of view, the security constraint is not satisfied because an input/output sequence (i.e., *Request2-Answer2.U-Request3-Answer3a.U*) can interfere with a higher input/output sequence (namely the insert in DB_S of the secret tuple $\langle Enterprise, Spying, Rigel \rangle$).

Generally, to avoid this kind of problem which occurs when we apply the non-interference model, the polyinstantiation technique is systematically employed. In particular, this is the case of the LDV project, which is based on the non-interference model. However, it is important to be able to consider that the existence of secret information is not always secret. As it is stated in [33], it would be better to consider that *"unless otherwise specified in a secrecy constraint, the system need not hide the existence of classified data in the database (PS#6: Security Policy Statement #6)"*. This means that polyinstantiation must not be automatically employed but only when explicitly specified in the security policy.

[4] This hypothesis is an integrity constraint used by most multilevel databases

1.2 Example 2: The elimination problem with non-deducibility

Let us now consider for this second example an extension of the previous example[5]. This extension concerns an additional relation, denoted *STR*, that indicates for each *Starship* its *Category*, that is to say its unique *Type* and its unique *Range* (as shown in Table 3).

Starship	Type	Range	TC
Enterprise U	Quick-and-light U	20000 U	U
Intergalactic U	Slow-and-heavy U	50000 U	U

Table 3. Two instances of the *STR* relation

Each attribute of this relation must not be obligatorily classified because it is only considered that the mission (i.e., *Objective* and *Destination*) of a starship can be confidential, but not its category (i.e., *Type* and *Range*).

Even if $user_U$ does not know the secret destination of *Enterprise*, from the knowledge of *Enterprise*'s range, he can nevertheless eliminate some of the originally possible destinations of *Enterprise* (see Figure 2). For instance, if Talos is a destination more than 20000 distant, then $user_U$ knows that $\neg Dest(Enterprise, Talos)$. From the non-deducibility point of view, the security constraint is not satisfied because $user_U$ can deduce that some destinations are impossible (i.e, he can eliminate some possible behavior of secret users).

Faced with this problem, a possible solution would be to change classifications, for instance to consider that $SR(Enterprise, 20000)$ is secret information. Nevertheless, to consider that $SR(Enterprise, 20000)$ is unclassified information may be necessary because this piece of information is related to the technical characteristics of Enterprise and is perhaps widely distributed and well-known information.

Actually, from the non-deducibility point of view, a confidential piece of information cannot be partially determined by unclassified information. However, it can be interesting to state that only a portion of this piece of information must be confidential (see for instance [7]). For example, only the confidentiality must be preserved for the two high-order bits, or for all the odd-order bits. The pertinence of such confidentiality constraints becomes obvious if they are applied to information such as the *employee's salary*. This approach can also be applied at a higher granularity level: at a byte level, or even at the level of the elementary pieces of information within a global and more complex data structure.

[5] A similar example was developed in [9]

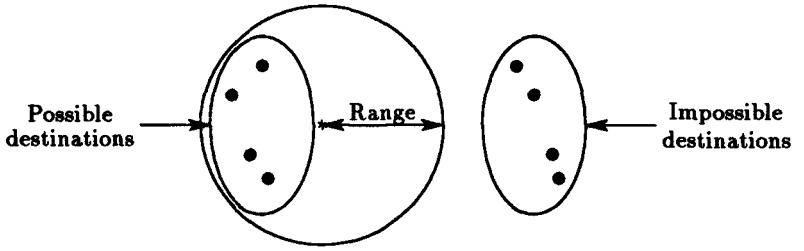

Fig. 2. *The possible/impossible destinations of Enterprise*

1.3 Synthesis

The examples of sections 1.1 and 1.2 illustrate that it is possible to differentiate several types of inference:

1. **Exact inference.** This inference occurs when a secret piece of information is exactly determined by a user whose clearance is unclassified.
2. **Partial inference.** It occurs when a user whose clearance is unclassified can reduce the set of possible values that can be assigned to a classified datum. This problem was first studied in [7]. Example of section 1.2 is an example of partial inference.
3. **Existential inference.** It occurs when a user whose clearance is unclassified derives the existence of a secret piece of information. Example of section 1.1 is an example of existential inference.

Most current research works focus on exact inference (for instance [35, 19, 17, 25]) and do not take into account the two other types of inference. We guess that these types of inference are equally important but are difficult to represent for at least two reasons :

1. To take into account these two types of inference, we need representing existential, disjunctive or negative information. Languages used in standard DBMS or in classical provers such as PROLOG do not provide these facilities. However, several researchers are currently developing this kind of extension (see for instance [12, 27, 30]).
2. It is clear that the exact inference of a secret piece of information is always a threat to confidentiality. Hence the security policy must prevent all exact inferences. On the other hand, every partial or existential inference does not necessarily represent a threat to confidentiality. For instance, let us assume that Paul's salary is equal to 10000 and this piece of information is classified at secret. Let us assume that an unclassified user can derive that this salary is between 9990 and 10010. It is a partial inference of Paul's salary and we can think that the security administrator will consider that this partial inference is not allowed. However, let us now assume that, in another situation, this

unclassified user can only infer that Paul's salary is greater that the SMIC[6]. It is another kind of partial inference of Paul's salary but it is clear that the security administrator cannot prevent from this partial inference.

Hence, to properly take into account partial and existential inferences, we must develop means which would allow the security administrator to precisely define which information is unclassified and which information is secret. It is only after doing so that we can distinguish acceptable states from unacceptable ones.

Notice that non interference and non deducibility properties respectively reject all existential inferences and partial inferences. This is the reason why we claim that these two security properties are too constraining to model database security.

2 A formal method to solve the different problems

When we want to correctly analyze problems we have to solve, it is important to come back to the concept of security policy for confidentiality. In this paper, we only consider the case of mandatory access control. An organization defines a mandatory access control policy which is applied to a set of sentences[7] $\mathcal{L}$ that represents the knowledge domain of the organization and a set of subjects $\mathcal{A}$ members of this organization. For each subject $A \in \mathcal{A}$, the policy divides the set of sentences $\mathcal{L}$ into two subsets:

1. The set of sentences $R(A)$ for which A is explicitly permitted to have an access.
2. The set of sentences $F(A)$ for which A is explicitly forbidden to have an access.

We do not assume that the policy is necessarily complete. This means that some sentences of $\mathcal{L}$ may not belong to either $R(A)$ or $F(A)$. For instance, let us take the example of the multilevel security policy. Some sentences $s \in \mathcal{L}$ receives a classification $l(s)$ and each subject $A \in \mathcal{A}$ receives a clearance $L(A)$. The sets $R(A)$ and $F(A)$ are then defined by:

$$R(A) = \{s \in \mathcal{L} \mid l(s) \leq L(A)\}$$

$$F(A) = \{s \in \mathcal{L} \mid \neg(l(s) \leq L(A))\}$$

In the following, we will assume that $R(A)$ is a consistent set of sentences. On the other hand, $F(A)$ is not necessarily consistent. We can then, as proposed by Dennings in [13], state two problems:

[6] The SMIC is the minimum salary allowed in France

[7] By a set of sentences, we mean a full first order language with a set of predicates, logical connectors (conjunction, disjunction, negation, implication) and existential and universal quantifiers.

1. Internal information flow controls.
2. Inference controls.

It is well known (see [29] for instance) that a computer system can be used to transmit information[8] not only by a direct access to a given piece of information but also by subtle and indirect means. Internal information flow controls are concerned with these leakages of information. To prohibit these leakages, we will show in section 3 that the system must control the permission to know any piece of information. This is a problem of knowledge conformity of an agent A with respect to its rights: A must only know pieces of information for which A has received a clearance.

When information derived from confidential data must be declassified for wider distribution, another leakage of information can occur: a user can use lower sensitive information stored in the database to which he can legally have an access to derive higher sensitive information. This leakage of information is called the inference problem. In this case, internal information controls as described above are not sufficient. Indeed, the flow of information is outside the computer system. Actually, we will show in section 4 that the inference problem does not occur if the security policy is correctly defined. This means we must verify, beside dividing $\mathcal{L}$, that the sets $R(A)$ and $F(A)$ are defined in a consistent manner (see Figure 3).

It is important to notice that the pioneering work of Bell and LaPadula deals essentially with internal flow controls and it does not take into account anymore the inference control. On the other hand, the majority of models designed to ensure information flow control (in particular non-interference and non-deducibility) jointly deal with the two problems.

The comparative study performed in [4] showed that these models do not provide a satisfactory control of internal flows of information. We also think that these models do not propose a satisfactory solution to inference control. Examples of sections 1.1 and 1.2 illustrate this point of view.

3 Internal information flow control

When reasoning about security, it is important to have a precise notion of what a subject **knows** and what a subject is **permitted to know**. Generally, what a subject A knows is represented by a modal operator denoted K_A. This modal operator was extensively studied and now has a well established semantics. We briefly recalled this semantics in section 1.

In our model, what a subject A is permitted to know is represented by another modal operator PK_A. This approach was first suggested by Glasgow and McEwen in [20]. In [4], we proposed a formal semantics for this modal operator. Intuitively, A is permitted to know φ if and only if A learns φ by playing the role of a user cleared to know the unclassified set of sentences $R(A)$. Hence, by identifying the authorized role of A with the unclassified set of sentences $R(A)$, we have:

$$PK_A\varphi \equiv K_{R(A)}\varphi$$

[8] By information, we mean a consistent subset of the language $\mathcal{L}$, i.e. a theory.

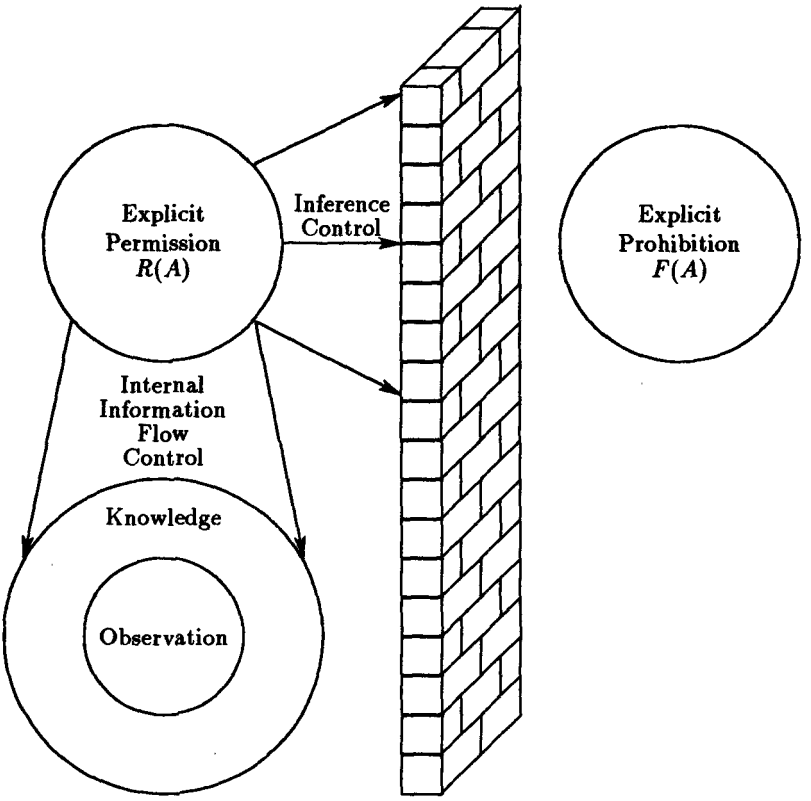

Fig. 3. The two problems to solve

A system is secure with respect to A if and only if the formula $K_A\varphi \rightarrow PK_A\varphi$ is valid (true in every traces of the system). This definition of security can be equivalently stated by the following requirement on the traces of the system:

Causality: For all traces t and t'

$$\text{If } t\lceil R(A) = t'\lceil R(A) \text{ then } t\lceil A = t'\lceil A$$

that is to say: the information A can observe should only depend on the information A is permitted to know. This means that causality rules out every non-authorized flow of information inside the computer system. In particular, causality controls every non-authorized indirect flows such as covert channels. In [4], we formally compare this definition of security with the classical definition of absence of information flow as non-interference and non-deducibility. It appears that causality has several advantages:

1. The explicit representation of time in the model proposed in [4] enables every covert channels to be controlled including timing channels.

2. Causality forces the system to be deterministic. This could appear as a drawback of the approach, however, it also enables every probabilistic covert channels to be controlled [9].
3. Causality does not include the tranquillity principle and provides efficient conditions to control the dynamic assignment of security levels and to perform secure downgradings [5].
4. It is possible to state the Brewer-Nash and Foley policies using the model of causality [8].
5. Causality has the hook-up property and we even propose an extension of this result in case of asynchronous composition [6]. This enables the security of a system to be analyzed in a modular way.
6. Causality does not require implicit assumptions on the subject's behavior. This enables the security of every non-input total systems to be analyzed [4].
7. Causality allows some kind of dependencies between unclassified information and secret information.

Thanks to these advantages (especially the two last points), causality does not rule out interesting system's behaviors, in particular those presented in sections 1.1 and 1.2. However, this can also lead to paradoxical examples in which a secure system copies high sensitivity inputs to low sensitivity outputs. These paradoxical situations exist when the information flow actually occurs externally, in the environment. Indeed, causality only provides a solution to the problem of internal information flow.

Similarly, let us analyze the security of the two examples proposed in sections 1.1 and 1.2. In both cases, the unclassified user, $user_U$, receives an answer computed by BD_U, and BD_U only contains information that $user_U$ is legally permitted to know. So, the output provided to $user_U$ only depends on unclassified information and, in that case, the internal flows of information are secure. Consequently, according to causality, these two examples are always secure.

As we have already suggested in section 1.3, the security administrator may specify that these two examples are actually insecure and he would consider in this case that the causality point of view is too optimistic. However, this only means that causality does not deal with the inference problem. In the following section, we show how to extend causality in order to take into account this problem.

4 Inference control

The inference problem in multilevel databases can be defined by the following: a user A can derive higher sensitive information, from lower sensitive information to which A has legally access. This problem seems a priori easy to solve because we can believe that it is sufficient to arbitrarily divide the set of relevant sentences $\mathcal{L}$ (those on which the multilevel security policy applies) into a set $R(A)$ of A's explicit permission and a set $F(A)$ of A's explicit prohibition. Actually, the

problem is much more tedious because each subject A can use its own permission to know some information but also general rules and common knowledge of the real world to derive new information (which are eventually forbidden). So, it must be verified, beside dividing $\mathcal{L}$, that the sets $R(A)$ and $F(A)$ were defined in a consistent manner. In order to correctly analyze this problem, we need a formal definition of the following concepts:

- The permission to know of each subject. We have already proposed a formal semantics for this concept for controlling internal flows of information. It is considered, with this semantics, that a subject A is permitted to know every given piece of information φ_0 for which A is explicitly permitted to have an access, i.e.:

  $$\text{If } \varphi_0 \in R(A) \text{ then } PK_A\varphi_0$$

  Moreover, A is implicitly authorized to perform valid derivation, i.e.:

  $$\text{If } PK_A\varphi \text{ and } PK_A(\varphi \rightarrow \psi) \text{ then } PK_A\psi$$

  that could be read: if A is permitted to know φ and if A is permitted to know $(\varphi \rightarrow \psi)$, then A is authorized to perform the derivation. Hence, A is permitted to know ψ.
- The prohibition to know of each subject, we denote it FK_A. We must formally define this concept. As in the case of the permission to know, the semantics of FK_A must enforce that a subject A is forbidden to know any given piece of information φ_0 for which A is explicitly forbidden to have an access, i.e.:

  $$\text{If } \varphi_0 \in F(A) \text{ then } FK_A\varphi_0$$

  Moreover, there exist implicit prohibitions[9], for instance:

  $$\text{If } FK_A\varphi \text{ or } FK_A\psi \text{ then } FK_A(\varphi \wedge \psi)$$

  that could be read: if A is forbidden to know φ or if A is forbidden to know ψ then A is implicitly forbidden to know the conjunction of φ and ψ. Notice, that the converse:

  $$\text{If } FK_A(\varphi \wedge \psi) \text{ then } FK_A\varphi \text{ or } FK_A\psi$$

  is generally not valid. For instance, think of the aggregation problem in which two pieces of information are more sensitive together than separate. The following sentence states another implicit prohibition:

  $$\text{If } FK_A(\varphi \vee \psi) \text{ then } FK_A\varphi \text{ and } FK_A\psi$$

[9] We refer to [10] for a detailed presentation of a complete set of such implicit prohibitions.

that could be read: if A is forbidden to know the disjunctive data $\varphi \vee \psi$, then A is also implicitly forbidden to know more informative data such as φ or ψ. This last axiom allows us to control a partial disclosure of information. For instance, let us assume that Paul's salary is equal to 10000 and that the security administrator actually specifies that the unclassified user A is forbidden to know that Paul's salary is between 8000 and 15000, that is to say:

$$FK_A(Salary(Paul, 8000) \vee ... \vee Salary(Paul, 15000))$$

Hence, in using the above axiom, A is also implicitly forbidden to reduce the set of values belonging to the interval [8000, 15000]. On the other hand, the converse of this axiom:

$$\text{If } FK_A\varphi \text{ and } FK_A\psi \text{ then } FK_A(\varphi \vee \psi)$$

is generally not valid. For instance, if the security administrator states that:

$$\forall sal, SMIC \leq sal \rightarrow FK_A Salary(Paul, sal)$$

then we cannot infer that:

$$FK_A(Salary(Paul, SMIC) \\ \vee \ Salary(Paul, SMIC + 1) \vee Salary(Paul, SMIC + 2) \vee ...)$$

that could be read: A is forbidden to know that Paul's salary is greater than the SMIC, which is common knowledge to any user in France.
Similarly, we do not consider that the following instance is an implicit prohibition:

$$\text{If } FK_A\varphi(c) \text{ then } FK_A(\exists x, FK_A\varphi(x)) \text{ (where } c \text{ is a given constant)}$$

This sentence could be read: if A is forbidden to know a secret piece of information $\varphi(c)$ (for instance $Destination(Enterprise, c)$), then A should not be always implicitly forbidden to know the existence of this secret piece of information. Notice that this assumption is made in the non-interference model. In our approach, it is the role of the security policy to explicitly specify the case for which the existence of a secret piece of information is also secret.

Notice also, that we do not consider that we have:

$$FK_A\varphi \equiv \neg PK_A\varphi$$

that could be read: A is forbidden to know φ if and only if A is not permitted to know φ. Indeed, we want to consider that there exists explicit permission to have an access (from which we derive what a subject is permitted to know) and explicit prohibition to have an access (from which we derive what a subject is forbidden to know). Generally, in reasoning about normative propositions, it is not assumed that $FK_A\varphi \equiv \neg PK_A\varphi$, especially when one wants to analyze the consistency of a given set of normative propositions (see [1] for instance). This is

exactly the case of the inference problem. This problem occurs when the security policy is not defined in a consistent way. This means that, if we want to avoid this problem, we must enforce the validity of the following sentence:

$$\neg(PK_A\varphi \wedge FK_A\varphi)$$

This sentence could be read: A cannot both have the permission to know φ and the prohibition to know φ. The enforcement of this condition guarantees that A cannot deduce forbidden information from permitted information by performing valid derivations. However, some difficulties remain because A can also perform plausible derivation. Through the following examples, we illustrate some of these problems and define possible solutions to solve them.

4.1 Example 3: Pseudo-consistency due to DB's incompleteness

When requests are sent to the secure DBMS that manages the *SOD* relation described in *Example 1*, answers that are formulated by this secure DBMS can depend on several parameters:

1. The clearance level of the user who sends the request to the DBMS (i.e., *Unclassified* or *Secret*).
2. The "characteristic" of the database (i.e., complete or incomplete).
3. The "characteristic" of the security policy that is applied to insure the internal information flow controls and, possibly, the inference controls.
4. The method that is used to avoid such inferences (i.e., with or without polyinstantiation).

It can thus be interesting to look at the inference problem in these different contexts, in order to be able to define some possible solutions to solve that inference problem. Let us consider that the security policy is defined so that $user_U$ is forbidden to know any secret piece of information and is also forbidden to know the existence of such secret information.

A first solution to enforce the security policy consistency is to consider that DB could be incomplete, this means that:

$$\exists star, [Starship(star) \in DB] \wedge$$
$$([(\exists obj, Objective(star, obj) \in DB) \wedge (\forall dest, Destination(star, dest) \notin DB)]$$
$$\vee [(\forall obj, Objective(star, obj) \notin DB) \wedge (\exists dest, Destination(star, dest) \in DB)]$$
$$\vee [(\forall obj, Objective(star, obj) \notin DB) \wedge (\forall dest, Destination(star, dest) \notin DB)])$$

When it is stated that $(\exists x, \forall y, Attribute(x, y) \notin DB)$, where *Attribute* means in the present example *Objective* or *Destination*, it is sometimes considered [27] that y is an undetermined value, denoted *Null* (see Table 4).

When DB is incomplete, the answer sent by the DBMS to $user_U$ for *Request3* is:

Request3: SELECT ⟨*Destination*⟩ FROM *SOD*
WHERE *Destination* = *Enterprise*
Answer3b.U: *Null*

Starship		Objective		Destination		TC
Enterprise	U	Spying	S	Rigel	S	S
Enterprise	U	Null	U	Null	U	U
Intergalactic	U	Exploration	U	Talos	U	U

Table 4. The multilevel *SOD* relation when *DB* is incomplete

This means that DB_U does not know the answer either, because there (perhaps) exists a response, in the real world, that *DB* does not know or, because there (actually) exists a response, in the real world and in *DB*, that $user_U$ cannot know because it is unknown by DB_U, and only known by DB_S.

From the assumption that *DB* could be incomplete, and by combining the *Answer2.U* and *Answer3b.U*, $user_U$ can deduce that:

$$K_{user_U}[(\exists dest, (Destination(Enterprise, dest) \in DB_S) \wedge (dest \neq Null)) \vee (\forall dest, Destination(Enterprise, dest) \notin DB)]$$

The incompleteness of *DB* could be considered as a satisfying approach from the security policy point of view because there is no inconsistency, since $user_U$ is not sure that there is a *Secret* destination for *Enterprise*; and, from the non-interference point of view this approach is really satisfying because there is *no* interference of DB_S on DB_U.

But if it is now considered that the probability, the possibility or the plausibility (let us more generally say the "certainty factor") of the fact $[\forall dest, Destination(Enterprise, dest) \notin DB]$ is very small, then there remains the eventuality that the security policy can be inconsistent, exactly as in the case previously studied with the *DB*'s completeness. Such a small certainty factor for the fact $[\forall dest, Destination(Enterprise, dest) \notin DB]$ could be obtained by $user_U$ with external information to *DB*, as it is stated in [18]. In fact, $user_U$ can suppose that the DBMS either does not know the real answer to *Request3*, or does not want to give him the response. He can go further in his reasoning by considering the second hypothesis, if he has access to external information, and thus build some abductive reasoning. Such external facts could be for examples:

- the fact that *Enterprise* will live soon, in a few days, and that the mission of any starship (and of course it's destination) must be known for some administrative reasons and thus recorded in the database (in DB_S in the present case) at least a few days before the date of the departure of the starship (let us say half a week);
- or the fact that $user_U$ has, or someone has for him, physically observed a great deal of activity all around *Enterprise*, signifying that this starship will live soon for a well-known mission;
- or also the fact that, usually, any starship does never stay idle more than a few days (let us say one week), and that *Enterprise* arrived five or six days ago.

To avoid such an abductive reasoning, another method could be used, *Polyinstantiation* (as it is often the case), but some management tools could nevertheless be very useful for the security manager to assess these certainty factors, for each fact such as: *"Enterprise will live soon"*, as it will be shown at the end of the next example.

4.2 Example 4: Pseudo-consistency due to *DB*'s polyinstantiation

Consider now that the database is complete again but polyinstantiated (as shown in Table 5):

$$\forall star \quad [\exists obj, \exists dest, SOD(star, obj, dest) \in DB_S] \\ \Rightarrow [\exists obj', \exists dest', SOD(star, obj', dest') \in DB_U \wedge \\ (obj' \neq obj) \wedge (dest' \neq dest)]$$

Starship		Objective		Destination		TC
Enterprise	U	Spying	S	Rigel	S	S
Enterprise	U	Exploration	U	Talos	U	U
Intergalactic	U	Exploration	U	Talos	U	U

Table 5. The multilevel polyinstantiated *SOD* relation

This means that each fact that exists in DB_S must also exist in DB_U, with a distinct value for each attribute. The security policy is the same as in *Example 3*, i.e.:

$FK_{user_U}[Destination(Enterprise, Rigel)] \wedge$
$FK_{user_U}[\exists dest, FK_{user_U} Destination(Enterprise, dest)]$

Therefore, the answer sent by DB_U to $user_U$ for *Request3* is:

Request3: SELECT ⟨*Destination*⟩ FROM *SOD*
WHERE *Starship* = *Enterprise*
Answer3c.U: ⟨*Talos*⟩

From the fact that *DB* is complete and polyinstantiated, $user_U$ can only be sure that:

$$\exists dest, Destination(Enterprise, dest) \in DB_U$$

However, if $user_U$ has access to contradictory external facts, he could suppose that another destination for *Enterprise* might exist, only known from DB_S. Hence, in the case of *DB*'s polyinstantiation, the abductive reasoning is still possible. Indeed, even if $user_U$, by some kind of external observation, can suppose that *Enterprise* will live soon (exactly as in the case where *DB* is incomplete),

he can nevertheless believe that the DBMS wants him to be mislead (see [18] for a more complete example).

It can thus be seen that some assistance tools may be very useful to the security manager for the assessment of the certainty factors of the different external facts that could be used for building some abductive reasoning. According to the external facts and their certainty factors, it is thus much easier to decide if any knowledge permitted to, or inferred by, $user_U$ is consistent with his explicit prohibitions.

4.3 The use of uncertainty to control abductive reasoning

As we have shown, respectively, in *Example 3* and *Example 4*, *DB*'s incompleteness and *DB*'s polyinstantiation are sometimes insufficient to avoid any effective information inference, in particular when abductive reasoning is possible. In both cases, we consider that uncertainty/certainty factors could be used to better appreciate such effective inference. Different types of uncertainty/certainty factors can be used (i.e., based, for example, on the possibility theory [38], or on the theory of evidence [32]); but, at the present step of our work, the following features of such uncertainty reasoning can be stated for any type of uncertainty/certainty factors:

- Uncertain and/or imprecise information can easily be represented in a linguistic form (which is very near to the natural language).
- The ordinal versus cardinal, and qualitative versus quantitative, aspects of uncertain and/or imprecise information are privileged.
- The mathematical frameworks that can be used, such as the possibility theory, are less normative than those classically used for representing certainty, such as probabilities.
- Uncertain and/or imprecise information can easily be combined, and thus updated, when represented in such a possibilistic framework.

This new type of approach used for representing, and evaluating the uncertainty/certainty of information stored in the system was already stated before (see for instance [36]) as a prospective research work for the assessment of the confidentiality preservation, referring in that case to the possibility theory [38, 16]. Independently, the same approach was also stated as an interesting solution to the abductive reasoning problem, in particular in relational DBMS [19, 18], referring in that case to the Dempster-Shafer theory [32].

So, when it is possible, only qualitative assessment is privileged faced to quantitative one. One advantage of such a representation is that it is much easier to combine distinct pieces of information and then compute the resulting certainty factor of the updated or resulting piece of information because the possibilistic or plausibilistic framework is less normative than the traditional probabilistic framework. This is mainly due to the following two reasons:

- The notion of independence of elementary events is not necessary to be able to compute the uncertainty/certainty factors of the piece of information issued from the combination of such elementary events.

- The fact that the sum of the certainty factors of two complementary events is not obligatorily equal to the unity (i.e., it is not always completely sure that either a given event, e, or its contrary, $\overline{e}$, will occur).

5 Conclusion

In this paper, we propose a general and complete method for modeling database security by splitting up the enforcement of a security policy into two subproblems:

- The internal information flow controls enforced via causality.
- The inference controls enforced via policy consistency.

We think that this decomposition provides a better understanding of the confidentiality problem. Notice that this method is not limited to the design of a secure database management system. Actually, it could apply to any information management system as soon as it is possible to give a formal correspondence between the objects stored in the system (such as tuples in the case of a relational DBMS) and formula in first order logic. However, much work remains to be done. First of all, we have formally defined the concept of prohibition to know some information using modal logic and possible world semantics. This formalism was presented in [10]. The aim is to build a tool the security manager could use to verify the consistency of the security policy he has defined. A possible scenario that could provide effective assistance for security managers could be the following:

1. The tool can prove that, a low user cannot, by using valid reasoning, derive higher sensitive information from lower sensitive information. We can conclude that no inference channel inside the security policy exists.
2. If no inference channel exists, then, for a given higher sensitive piece of information, the tool can use abductive reasoning to find what information α the low subject needs to assume in order to derive this higher sensitive piece of information.
3. The tool asks the security manager whether it is plausible that a low subject might assume α. In a more ambitious scenario, the tool can use plausible reasoning in order to put itself in the position of the database security manager who tries to answer this question.

Finally, notice that, in this paper, we focus on mandatory access control and we do not consider discretionary access controls. It could be interesting to see whether the method we proposed can be applied to both kinds of access (mandatory and discretionary).

Acknowledgement

We would like to thank the DRET for its support, Jill Manning for her help, and the anonymous referees for their comments on a previous draft of this paper.

References

1. C. E. Alchourron. Philosophical Foundations of Deontic Logic and its Practical Applications in Computational Contexts. In *Proc. of the First International Workshop on Deontic Logic in Computer Science*, Amsterdam, The Netherlands, 1991. Invited Lecture.
2. D. Bell and L. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR-2997, MITRE, Bedford, Mass, 1975.
3. P. Bieber and F. Cuppens. Computer Security Policies and Deontic Logic. In *Proc. of the First International Workshop on Deontic Logic in Computer Science*, Amsterdam, The Netherlands, 1991.
4. P. Bieber and F. Cuppens. A Logical View of Secure Dependencies. *Journal of Computer Security*, 1(1):99–129, 1992.
5. P. Bieber and F. Cuppens. Secure Dependencies with Dynamic Level Assignments. In *Proc. of the computer security foundations workshop*, Franconia, 1992.
6. N. Boulahia-Cuppens and F. Cuppens. Asynchronous composition and required security condition. In *IEEE Symposium on Security and Privacy*, Oakland, 1994.
7. E. Cohen. Information Transmission in Sequential Programs. In *Foundations of Secure Computation*. Academic Press, 1978.
8. F. Cuppens. A modal logic framework to solve aggregation problems. In S. Jajodia and C. Landwehr, editors, *Database Security, 5: Status and Prospects*. North-Holland, 1992. Results of the IFIP WG 11.3 Workshop on Database Security.
9. F. Cuppens. A Logical Analysis of Authorized and Prohibited Information Flows. In *IEEE Symposium on Security and Privacy*, Oakland, 1993.
10. F. Cuppens and R. Demolombe. Normative Conflicts in a Confidentiality Policy. In *ECAI-94 Workshop on Artificial Normative Reasoning*, Amsterdam, The Netherlands, 1994.
11. F. Cuppens and K. Yazdanian. A "Natural" Decomposition of Multi-level Relations. In *IEEE Symposium on Security and Privacy*, Oakland, 1992.
12. R. Demolombe and L. Fariñas del Cerro. Efficient representation of incomplete information. In J. Schmidt and C Thanos, editors, *Foundations of Knowledge Base Management*. Springer Verlag, 1990.
13. D. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.
14. D. Denning, T. Lunt, R. Shell, M. Heckman, and W. Shockley. A Multilevel Relational Data Model. In *IEEE Symposium on Security and Privacy*, Oakland, 1987.
15. D. Denning, T. Lunt, R. Shell, W. Shockley, and M. Heckman. The SeaView Security Model. In *IEEE Symposium on Security and Privacy*, Oakland, 1988.
16. D. Dubois and H. Prade. *Possibility Theory: an approach to computerized processing of uncertainty*. Plenum Press, 1988.
17. T. Garvey, T. Lunt, X. Qian, and M. Stickel. Toward a Tool to Detect and Eliminate Inference Problems in the Design of Multilevel Databases. In *Proc. of the Sixth IFIP WG 11.3 Working Conference on Database Security*, Vancouver, 1992.
18. T. D. Garvey and T. F. Lunt. Cover Stories for Database Security. In S. Jajodia and C. Landwehr, editors, *Database Security, 5: Status and Prospects*. North-Holland, 1992. Results of the IFIP WG 11.3 Workshop on Database Security.
19. T. D. Garvey, T. F. Lunt, and M. E. Stickel. Abductive and Approximate Reasoning Models for Characterizing Inference Channels. In *Proc. of the computer security foundations workshop*, Franconia, 1991.

20. J. Glasgow and G. McEwen. Reasoning about knowledge and permission in secure distributed systems. In *Proc. of the computer security foundations workshop*, Franconia, 1988.
21. J. Goguen and J. Meseguer. Unwinding and Inference Control. In *IEEE Symposium on Security and Privacy*, Oakland, 1984.
22. M. J. Grohn. A model of a protected data management system. Technical Report ESD-TR-76-289, I. P. Sharp Associates Ltd., Bedford, Mass, 1976.
23. J. Guttman and M. Nadel. What needs securing. In *Proc. of the computer security foundations workshop*, Franconia, 1988.
24. J. T. Haigh, R. C. O'Brien, P. D. Stachour, and D. L. Toups. The LDV Approach to Database Security. In D. L. Spooner and C. Landwehr, editors, *Database Security, III: Status and Prospects*. North-Holland, 1990. Results of the IFIP WG 11.3 Workshop on Database Security.
25. T. H. Hinke. Inference Aggregation Detection in Database Management Systems. In *IEEE Symposium on Security and Privacy*, Oakland, 1988.
26. T. H. Hinke and M. Schaeffer. Secure data management system. Technical Report RADC-TR-75-266, System Development Corporation, 1975.
27. T. Imielinski and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4), October 1984.
28. S. Jajodia and R. Sandhu. Polyinstatiation Integrity in Multilevel Relations. In *IEEE Symposium on Security and Privacy*, Oakland, 1990.
29. B. W. Lampson. A note on the confinement problem. *Communication of the Association for Computing Machinery*, 16(10):613–615, 1973.
30. K.-C. Liu and R. Sunderraman. General indefinite and maybe information in relational databases. In R. Ritter, editor, *Information processing 89*, pages 809–814, New-York, 1989. Elsevier.
31. R. Sandhu and S. Jajodia. Honest Databases That Can Keep Secrets. In *Proceedings of the 14th National Computer Security Conference*, Washington, D.C., 1991.
32. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
33. G. W. Smith. Multilevel Secure Database Design: A Practical Application. In *Fifth Annual Computer Security Applications Conference*, Tucson, Arizona, 1989.
34. D. Sutherland. A Model of Information. In *Proceedings of the 9th National Computer Security Conference*, 1986.
35. B. Thuraisingham, W. Ford, M. Collins, and J. O'Keefe. Design and implementation of a database inference controller. *Data & Knowledge Engineering*, 11(3), December 1993.
36. G. Trouessin. Quantitative Evaluation of Confidentiality by Entropy Calculation. In *Proc. of the computer security foundations workshop*, Franconia, 1991.
37. R. A. Whitehurst and T. F. Lunt. The Seaview Verification. In *Proc. of the computer security foundations workshop*, Franconia, 1989.
38. L. A. Zadeh. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1, 1978.