A full formalisation of the Bell and La Padula security model

E. Gureghian, Th. Hardin, M. Jaume*

Abstract

Information access control programs are based on a security policy model. Flaws in them may come from a lack of precision or some incoherences in the policy model or from inconsistencies between the model and the code. In this paper, we build a full mechanized formalization of the well-known Bell and LaPadula policy model, checking all the steps of the proofs. Then, we derive automatically a program for the access controls considered in this model. Such a program implements a transition function which has been formally proved sound according to the three security properties involved in the Bell and La Padula model. All the work is done within Coq, a theorem prover based on a very strong type theory.

Keywords: security policy, Bell and La Padula model, formal methods

1 Introduction

Security of information systems has becomed a major problem of our societies and a well-etablished field of computer science. Research themes involve access control mechanisms, modeling of information flow and its applications to confidentiality policies [10], mobile code security, cryptographic protocols, etc. The methods to consider these questions are evolving, just at the ones used in safety area, where ad-hoc and empirical approachs were progressively replaced by more formal methods. High levels of safety require that the requirement/specification phase is done using mathematical models, allowing to have mechanized proofs of the required properties. In the same way, insurance in the security of systems asks for the use of true formal methods along the process of software development, starting at the specification level.

Computer information security is usually seen as the combination of three classes of properties: confidentiality (denying unauthorised accesses), integrity (denying unauthorised modifications of information) and availability (denying unauthorised uses of ressources). To promote the conception of trusted systems, security evaluation criterias have been elaborated by governments agencies, for example the Trusted Computer Security Evaluation Criterias (1983) (TCSEC), the Information Technology Security Evaluation Criterias (1991) (ITSEC) and the Common Criterias (1999), which are a collection of normative documents. These criterias provide both a framework for the software industry to ensure that software has been carefully designed and a referential for its customers. A product evaluation and certification against the common criterias framework is built according to two hypotheses. The first one is the "protection profile", that is, under which conditions the evaluated product is supposed to be used. The second one is its level of assurance, which is simply the level of trust the system can receive according to the way it was developed. A good security level can be reached if the product is evaluated at an assurance level greater than EAL-5

*SPI-LIP6, University Paris 6, France

against a convincing protection profile such as the one given in [13]. Such a profile requires the use of a mandatory formal policy model in order to achieve ab acceptable level of confidentiality. Mandatory access control (MAC), contrary to discretionnary access controls (DAC) such as access control list (ACL), is managed and enforced by the underlying system rather than by an authorized user.

Such a policy model must be precise and unambiguous and thus must be described in a mathematical formalism. This is the case for the Bell and La Padula [2] model. It is exposed in four successive refinements: the mathematical foundations, the mathematical model, a refinement of the mathematical model and a unified exposition together with a MULTICS interpretation [3]. Note that such a model is only concerned with confidentiality. Nevertheless, the Biba integrity formal model is very close to the Bell and LaPadula one and is also considered in the same protection profile[13].

Having a mathematical model drawn by hand is a very serious way to increase confidence. But, this is not enough. Attempts to check proofs done by hand with a theorem prover has thrawn away a lot of them. Often, the errors are introduced by points considered as evident details or by forgotten cases. In this paper, we give a formal description of Bell and LaPadula model, checked by the theorem prover Coq [15]. From this specification, an implementation is automatically extracted and thus fully certified. It is the transition function of an abstract state machine.

In this paper, we expose the model, the code and the proofs. However, as all readers are perhaps not very acquainted with Coq, we describe the specification and the proofs with a quite usual mathematical language, in fact very close to Coq syntax. The complete implementation is available on the site `http://www-spi.lip6.fr/~jaume/`. In the following, we first give a short presentation of Coq, which is a higher-order logic proof assistant implementing the calculus of inductive constructions [7] and which alows to extract programs from proofs [14]. Then , we give a full description of the Bell and LaPadula model and of its implementation. We assume the reader familiar with the Bell and La Padula model as presented in [2]. A more general discussion on it together with a survey on security models can be found in [12].

2 A (very) brief vescription of Coq

We use here version 7.3 of the proof assistant Coq which allows the interactive development of formal proofs. In order to make this paper more readable, we adopt here a pseudo-Coq syntax which differs slightly from the usual Coq syntax. The Coq tool is based on a logical framework known as the calculus of inductive constructions, which is an extension of a typed λ-calculus supporting dependent types, polymorphic types, and type constructors. The basic idea underlying this logical framework, based on the Curry-Howard isomorphism, is that a proof of a proposition can be seen as a functional object. For instance, a proof of a proposition of the form $A \Rightarrow B$ is a function mapping every proof of A to a proof of B. The type of this function is isomorphic to the proved proposition, so types and proposition are identified, as are objects and proofs. Furthermore, this framework allows the definition of inductive and co-inductive types (which are specified by their constructors). Constructing a proof within Coq is an interactive process: given a goal, the user specifies which deduction rule should be applied, and Coq does all the "computations". The theorem prover solves successive subgoals with tactics (i.e., functions that build a proof of a given goal from proofs of more elementary subgoals). In all our development, we use a Coq package for finite sets implemented as lists: we will write $[\![A]\!]$ the type of finite sets of elements of type A, over which classical operations on sets, such as $\in$, $\subseteq$, $\cup$, ..., are defined. Given two terms E_1 and E_2 of type $[\![A]\!]$, we will write $E_1 \equiv E_2$ as a shortcut of $E_1 \subseteq E_2 \wedge E_2 \subseteq E_1$. In order to check if every element of a finite set satisfies a (decidable) property, we define a function $\mathsf{for\_all}$ of type $[\![A]\!] \to (A \to \mathbb{B}) \to \mathbb{B}$ where A is any type needed and $\mathbb{B}$ is the inductive set of booleans. Similarly, we define a function filter of type $[\![A]\!] \to (A \to \mathbb{B}) \to [\![A]\!]$ whose result is the set of elements of a given set which satisfy a (decidable) property. It is important to note that we write $t : T$ to express that the type of a term t is T, while we write $t \in E$ to express that a term t of type T belongs to the set E of type $[\![T]\!]$ (thus the term $t \in E$ is of type $\mathbb{B}$).

3 System representation

The system is represented by an abstract machine containing a state that operations (or requests) can change.

3.1 Parameters and hypothesis

The Bell and La Padula security model describes a set of access control rules between active entities, called subjects (representing processes, programs in execution ...) and passive entities, called objects (representing data, files, programs, subjects, I/O devices ...). Each subject and object is associated with two independent informations: the set of *needs-to-know* (i.e., a special access privileges to a subject, the topic of the data) and the *classification* (i.e., the clearance level of a subject, the sensibility of the data) We first define $\mathcal{O}$ and $\mathcal{S}$ as sets of objects and subjects. In order to be able to enumerate elements belonging to these sets, we suppose $\mathcal{O}$ and $\mathcal{S}$ to be countable sets, by indexing them by natural numbers. Then, we introduce a set $\mathcal{K}$ of needs-to-know and a set $\mathcal{C}$ of classifications as variables, thus these sets are parameters of our development. Furthermore, we assume that equality over these sets is decidable. We alse introduce as an assumption a total order relation $\preceq$ over $\mathcal{C}$.

```
Definition 𝒪 :  Set := ℕ.        Definition 𝒮 :  Set := ℕ.
Variable 𝒦 :  Set.
Hypothesis eq_dec𝒦 :  ∀c1, c2 : 𝒦   {c1 = c2} + {c1 ≠ c2}.
Variable 𝒞 :  Set.                Variable ⪯: 𝒞 → 𝒞 → 𝔹.
Hypothesis eq_dec𝒞 :  ∀c1, c2 : 𝒞   {c1 = c2} + {c1 ≠ c2}.
Hypothesis Total_order :  ∀c1, c2 : 𝒞   {c1 ⪯ c2} + {c2 ⪯ c1}.
Hypothesis ReflC : ∀c : 𝒞   c ⪯ c.
Hypothesis ASymC : ∀c1, c2 : 𝒞   c1 ⪯ c2 ⇒ c2 ⪯ c1 ⇒ c1 = c2.
Hypothesis TransC : ∀c1, c2, c3 : 𝒞   c1 ⪯ c2 ⇒ c2 ⪯ c3 ⇒ c1 ⪯ c3  .
```

(+ denotes the disjunction.)

The *security level* associated with an object or a subject can be seen as a pair (c, k) where c is a classification and k is a set of needs-to-know. Given two entities e_1 and e_2 respectively associated with the security levels (c_1, k_1) and (c_2, k_2), we will say that e_1 *dominates* e_2 if $c_2 \preceq c_1$ and if k_1 is a superset of k_2. Note that the relation "dominates" is a partial order.

3.2 States

A state is a pair (m, f) where m is a matrix containing current accesses and access rights and f is a classification vector. Thus, we define the type of states as the following product:

```
Definition Σ:Set := ℳ × ℱ.
```

where $\mathcal{M}$ and $\mathcal{F}$ are defined in the two following subsections.

3.2.1 Current accesses and access rights

We define $\mathcal{A}$ as the set of access attributes containing the following elements: `r` for "read-only" (i.e., "pure read"), `w` for "read-write", `e` for "execute", `a` for "append" (i.e., "pure-write"), and `c` for "control" (allowing to update control accesses).

```
Inductive 𝒜:Set := r:𝒜 | w:𝒜 | e:𝒜 | a:𝒜 | c:𝒜.
```

Current accesses and access rights are usually represented by two matrices. One way to define these notions within Coq is to define the type $\mathcal{M}$ as follows:

```
Record ℳ : Set := mkM {
  M : 𝒮 → 𝒪 → ⟦𝒜⟧ × ⟦𝒜⟧ ;   Nₒ : ℕ ;   Nₛ : ℕ }.
```

Given a term m of type $\mathcal{M}$, m will be called a matrix and we will write $m.M$ (resp. $m.N_o$, $m.N_s$) to denote the field M (resp. N_o, N_s) of m. Since, we require the number of subjects and objects to be finite (this requirement is needed in order to be able to enumerate elements of the matrix in a finite way), we introduce two fields N_o and N_s corresponding respectively to the number of objects and the number of subjects minus 1 since both the subjects and the objects of the matrix are indexed from 0. These fields are used to define an access function $A_{\mathcal{M}}$ to the elements of a matrix m of type $\mathcal{M}$:

$$\begin{array}{l} A_{\mathcal{M}} : \mathcal{M} \rightarrow \mathcal{S} \rightarrow \mathcal{O} \rightarrow [\![\mathcal{A}]\!] \times [\![\mathcal{A}]\!] \\ := \lambda m : \mathcal{M}.\lambda s : \mathcal{S}.\lambda o : \mathcal{O}. \left\{ \begin{array}{ll} m.M(s,o) & \text{if } s \leq m.N_s \text{ and } o \leq m.N_o \\ (\emptyset, \emptyset) & \text{otherwise} \end{array} \right. \end{array}$$

where $\lambda x : T.e$ denotes the function whose argument is a term x of type T and whose body is the term e. In the following we will use the following notations:

- we will write $m_{(1)}[s,o]$ for the first projection of $A_{\mathcal{M}}(m.M, s, o)$, representing the current accesses of the subject s over the object o
- we will write $m_{(2)}[s,o]$ for the second projection of $A_{\mathcal{M}}(m.M, s, o)$, representing the access rights of the subject s over the object o

Given a matrix m of type $\mathcal{M}$, an object o is said to be *opened* according to the right $\alpha : \mathcal{A}$, if there is a subject s such that $\alpha \in m_{(1)}[s,o]$. Similarly, a subject s is said to be *granted* the right α over an object o if $\alpha \in m_{(2)}[s,o]$. Since, as we said, elements of m can be enumerated, we define a function Ω of type $\mathcal{M} \rightarrow \mathcal{S} \rightarrow [\![\mathcal{A}]\!] \rightarrow [\![\mathcal{O}]\!]$, such that $\Omega(m, s, E)$ contains all the objects opened by the subject s according to an access mode α in E. The following property is easily proved:

$$\begin{array}{l} \forall m : \mathcal{M}\ \forall s : \mathcal{S}\ \forall \alpha_1 : \mathcal{A}\ \forall \alpha_2 : \mathcal{A}\ \forall o : \mathcal{O} \\ \quad o \in \Omega(m, s, \{\alpha_1, \alpha_2\}) \Rightarrow (o \in \Omega(m, s, \{\alpha_1\}) \vee o \in \Omega(m, s, \{\alpha_2\})) \end{array} \tag{1}$$

The following operations over matrices are now defined. We first define a function $\oplus_1 : \mathcal{M} \rightarrow \mathcal{S} \rightarrow \mathcal{O} \rightarrow \mathcal{A} \rightarrow \mathcal{M}$ allowing to add an access $x : \mathcal{A}$ in the current accesses of a matrix $m : \mathcal{M}$ for a subject $s : \mathcal{S}$ over an object $o : \mathcal{O}$. We will write $m \oplus_1 \langle s, o, x \rangle$ the matrix obtained. Of course, only the field M is updated (i.e., the fields N_o and N_s are not modified):

$$\begin{array}{ll} & M.(m \oplus_1 \langle s, o, x \rangle) \\ := & \lambda s' : \mathcal{S}.\lambda o' : \mathcal{O}. \left\{ \begin{array}{ll} (\{x\} \cup m_{(1)}[s,o], m_{(2)}[s,o]) & \text{if } s = s' \text{ and } o = o' \\ (m_{(1)}[s',o'], m_{(2)}[s',o']) & \text{otherwise} \end{array} \right. \end{array}$$

Similarly, we define a function $\oplus_2 : \mathcal{M} \rightarrow \mathcal{S} \rightarrow \mathcal{O} \rightarrow \mathcal{A} \rightarrow \mathcal{M}$ allowing to add an access right in a matrix.

$$\begin{array}{ll} & M.(m \oplus_2 \langle s, o, x \rangle) \\ := & \lambda s' : \mathcal{S}.\lambda o' : \mathcal{O}. \left\{ \begin{array}{ll} (m_{(1)}[s,o], \{x\} \cup m_{(2)}[s,o]) & \text{if } s = s' \text{ and } o = o' \\ (m_{(1)}[s',o'], m_{(2)}[s',o']) & \text{otherwise} \end{array} \right. \end{array}$$

Deleting a current access or an access right is defined in the same way by the two functions $\ominus_1$ and $\ominus_2$ of type $\mathcal{M} \to \mathcal{S} \to \mathcal{O} \to \mathcal{A} \to \mathcal{M}$ defined as follows:

$$\begin{array}{rl} & M.(m \ominus_1 \langle s, o, x\rangle) \\ := & \lambda s' : \mathcal{S}.\lambda o' : \mathcal{O}. \begin{cases} (m_{(1)}[s,o]\backslash\{x\}, m_{(2)}[s,o]) & \text{if } s = s' \text{ and } o = o' \\ (m_{(1)}[s',o'], m_{(2)}[s',o']) & \text{otherwise} \end{cases} \end{array}$$

$$\begin{array}{rl} & M.(m \ominus_2 \langle s, o, x\rangle) \\ := & \lambda s' : \mathcal{S}.\lambda o' : \mathcal{O}. \begin{cases} (m_{(1)}[s,o], m_{(2)}[s,o]\backslash\{x\}) & \text{if } s = s' \text{ and } o = o' \\ (m_{(1)}[s',o'], m_{(2)}[s',o']) & \text{otherwise} \end{cases} \end{array}$$

We also define a function $\circleddash : \mathcal{M} \to \mathcal{S} \to \mathcal{O} \to (\llbracket\mathcal{A}\rrbracket \times \llbracket\mathcal{A}\rrbracket) \to \mathcal{M}$ allowing to assign a value to $m[s,o]$

$$\begin{array}{rl} & M.(m \circleddash \langle s, o, (E_1, E_2)\rangle) \\ := & \lambda s' : \mathcal{S}.\lambda o' : \mathcal{O}. \begin{cases} (E_1, E_2) & \text{if } s = s' \text{ and } o = o' \\ (m_{(1)}[s',o'], m_{(2)}[s',o']) & \text{otherwise} \end{cases} \end{array}$$

Last, we define a function $\circledin : \mathcal{M} \to \mathcal{O} \to \mathbb{B}$ such that, given a matrix $m : \mathcal{M}$ and an object $o : \mathcal{O}$, $o \circledin m$ is `false` if and only if for all subject $s \leq N_s$, $m_{(2)}[s,o] = \emptyset$. In other words, since $m_{(2)}[s,o]$ denotes the access rights over o, we have $o \circledin m$ if for at least one subject s, the object o can be accessed. Such objects are called *live* objects[1].

We prove that opened objects by a subject $s_2 : \mathcal{S}$, according to a set of access attributes $E : \llbracket\mathcal{A}\rrbracket$, are exactly the same if either we add an access attribute for a subject $s_1 \neq s_2$ or we add an access attribute $x \notin E$ to the current accesses:

$$\begin{array}{l} \forall m : \mathcal{M}\ \forall s_1, s_2 : \mathcal{S}\ \forall o : \mathcal{O}\ \forall x : \mathcal{A}\ \forall E : \llbracket\mathcal{A}\rrbracket \\ \quad (s_1 \neq s_2) \vee (x \notin E) \Rightarrow \Omega(m \oplus_1 \langle s_1, o, x\rangle, s_2, E) = \Omega(m, s_2, E) \end{array} \tag{2}$$

We also prove that, given the matrix $m \oplus_1 \langle s_1, o_1, x\rangle$, if an object o_2 is opened by a subject s_2 according to a set of access attributes E, then either $o_1 = o_2$ or o_2 was already opened in m:

$$\begin{array}{l} \forall m : \mathcal{M}\ \forall s_1, s_2 : \mathcal{S}\ \forall o_1, o_2 : \mathcal{O}\ \forall x : \mathcal{A}\ \forall E : \llbracket\mathcal{A}\rrbracket \\ \quad o_2 \in \Omega(m \oplus_1 \langle s_1, o_1, x\rangle, s_2, E) \Rightarrow o_1 = o_2 \vee o_2 \in \Omega(m, s_2, E) \end{array} \tag{3}$$

Furthermore, we prove the following property over $\ominus_1$:

$$\forall m : \mathcal{M}\ \forall s : \mathcal{S}\ \forall s' : \mathcal{S}\ \forall o : \mathcal{O}\ \forall o' : \mathcal{O}\ \forall \alpha : \mathcal{A} \quad (m \ominus_1 \langle s, o, \alpha\rangle)_{(1)}[s',o'] \subseteq m_{(1)}[s',o'] \tag{4}$$

Thus, it follows:

$$\forall m : \mathcal{M}\ \forall s : \mathcal{S}\ \forall s' : \mathcal{S}\ \forall o : \mathcal{O}\ \forall E : \llbracket\mathcal{A}\rrbracket\ \forall \alpha : \mathcal{A} \quad \Omega(m \ominus_1 \langle s, o, \alpha\rangle, s', E) \subseteq \Omega(m, s', E) \tag{5}$$

3.2.2 Classification vectors

Each object and subject possesses a classification and a finite set of needs-to-know. Thus, we define the type $\mathcal{F}$ as follows:

Record $\mathcal{F}$: Set := mkF {
$f_1 : \mathcal{S} \to \mathcal{C}$; $f_2 : \mathcal{O} \to \mathcal{C}$; $f_3 : \mathcal{S} \to \llbracket\mathcal{K}\rrbracket$; $f_4 : \mathcal{O} \to \llbracket\mathcal{K}\rrbracket$ }

[1] In the original paper of Bell and La Padula, the set $\mathcal{A}(m)$ of live objects is defined and is only used to test if an object o belongs to it. Hence, since the construction of this set is computationally very expensive, we only implement the function $\circledin$ which stops as soon as it finds a subject s such that $m_{(2)}[s,o] \neq \emptyset$.

A term f of type $\mathcal{F}$ is called a classification vector, and we will write it $(\varphi_1, \varphi_2, \varphi_3, \varphi_4)$ where φ_1 (resp. φ_2, φ_3 and φ_4) stands for $f.f_1$ (resp. $f.f_2$, $f.f_3$ and $f.f_4$).

As we did for matrices, given an object $o : \mathcal{O}$, a classification $c : \mathcal{C}$ and a set of needs-to-know $E : [\![\mathcal{K}]\!]$, we define a function $\oslash : \mathcal{F} \to \mathcal{O} \to \mathcal{C} \to [\![\mathcal{K}]\!] \to \mathcal{F}$ over a vector classification f allowing to set new values to $\varphi_2(o)$ and $\varphi_4(o)$ (note that no modification is allowed for the functions φ_1 and φ_3):

$$(\varphi_1, \varphi_2, \varphi_3, \varphi_4) \oslash \langle o, c, E \rangle = (\varphi_1, \varphi_2', \varphi_3, \varphi_4')$$

where φ_2' and φ_4' are defined as follows:

$$\varphi_2' := \lambda o' : \mathcal{O}. \begin{cases} c & \text{if } o = o' \\ \varphi_2(o) & \text{otherwise} \end{cases} \qquad \varphi_4' := \lambda o' : \mathcal{O}. \begin{cases} E & \text{if } o = o' \\ \varphi_4(o) & \text{otherwise} \end{cases}$$

3.2.3 Security properties over states

We can now define the security properties over states that are considered in the Bell and La Padula model. We focus here on two types of access control policies: *Discretionnary Access Control* (DAC) consisting in the control of access rights (based on file ownership), and *Mandatory Access Control* (MAC) which restricts how users can pass rights to other users (such a policy is motivated by the existence of programs known as *Trojan Horses*[2]).

Let $\rho = (m, (\varphi_1, \varphi_2, \varphi_3, \varphi_4))$ be a state. We define the three following security properties over ρ. The type of DAC, MAC and MAC* is $\Sigma \to \texttt{Prop}$.

The DAC property states that current accesses must always belong to the set of authorized accesses. So, we define $\mathsf{DAC}(\rho)$ as the following proposition:

$$\forall s : \mathcal{S}\ \forall o : \mathcal{O} \quad m_{(1)}[s, o] \subseteq m_{(2)}[s, o]$$

Of course, this property remains true when adding an access right in the matrix. Indeed, it is easy to prove that:

$$\forall \rho = (m, f) : \Sigma\ \forall s : \mathcal{S}\ \forall o : \mathcal{O}\ \forall \alpha : \mathcal{A} \quad \mathsf{DAC}(\rho) \Rightarrow \mathsf{DAC}((m \oplus_2 \langle s, o, \alpha \rangle, f)) \tag{6}$$

The MAC property states the no "read-up" property: no subject can gain read access over an object whose classification is higher than its classification or whose set of needs-to-know is not included in its set of needs-to-know. Since $\mathtt{w}$ is a "read/write" attribute, both $\mathtt{r}$ and $\mathtt{w}$ accesses must be enforced. We define $\mathsf{MAC}(\rho)$ as the following proposition:

$$\forall s : \mathcal{S}\ \forall o : \mathcal{O}\ \ (\mathtt{r} \in m_{(1)}[s, o] \vee \mathtt{w} \in m_{(1)}[s, o]) \Rightarrow (\varphi_2(o) \preceq \varphi_1(s) \wedge \varphi_4(o) \subseteq \varphi_3(s))$$

Here again, this property remains true when adding an access right in the matrix:

$$\forall \rho = (m, f) : \Sigma\ \forall s : \mathcal{S}\ \forall o : \mathcal{O}\ \forall \alpha : \mathcal{A} \quad \mathsf{MAC}(\rho) \Rightarrow \mathsf{MAC}((m \oplus_2 \langle s, o, \alpha \rangle, f)) \tag{7}$$

[2] An example of *Trojan Horses* program is a program giving (high-level) rights associated with the user which executes it to the owner of this program which is associated with low security level.

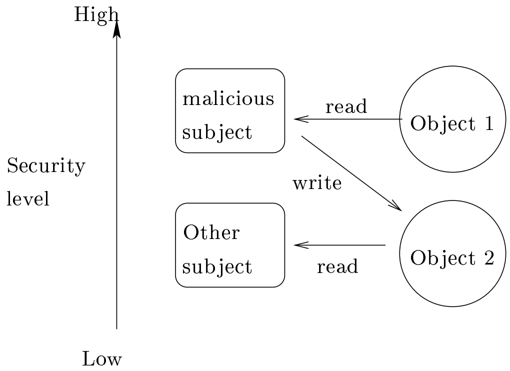

Figure 1: Violation of the $\mathsf{MAC}^\star$ property

The $\mathsf{MAC}^\star$ property corresponds to the no "write-down" property: it is prohibited for any subject currently accessing an object o_2 in a read-equivalent mode (i.e., $\mathtt{r}$ or $\mathtt{w}$) to access in a write-equivalent mode (i.e., $\mathtt{a}$ or $\mathtt{w}$) any object o_1 such that $\varphi_2(o_2) \not\preceq \varphi_2(o_1)$ or $\varphi_4(o_2) \not\subseteq \varphi_4(o_1)$:

$$\begin{array}{l} \forall s : \mathcal{S}\ \forall o_1 \in \Omega(m, s, \{\mathtt{w}, \mathtt{a}\})\ \forall o_2 \in \Omega(m, s, \{\mathtt{r}, \mathtt{w}\}) \\ \quad \varphi_2(o_2) \preceq \varphi_2(o_1) \wedge \varphi_4(o_2) \subseteq \varphi_4(o_1) \end{array}$$

This property prevents copying of an object to a lower security level by a "malicious" subject (see figure 1). Hence, we define $\mathsf{MAC}^\star(\rho)$ as the following proposition:

$$\begin{array}{l} \forall s : \mathcal{S} \\ \quad \mathsf{for\_all}(\quad \Omega(m, s, \{\mathtt{w}, \mathtt{a}\}), \\ \qquad\qquad \lambda o_1 : \mathcal{O}.\mathsf{for\_all}(\quad \Omega(m, s, \{\mathtt{r}, \mathtt{w}\}), \\ \qquad\qquad\qquad \lambda o_2 : \mathcal{O}.(\varphi_2(o_2) \preceq \varphi_2(o_1) \wedge \varphi_4(o_2) \subseteq \varphi_4(o_1)))) \end{array}$$

This property also remains true when adding an access right in the matrix:

$$\forall \rho = (m, f) : \Sigma\ \forall s : \mathcal{S}\ \forall o : \mathcal{O}\ \forall \alpha : \mathcal{A} \quad \mathsf{MAC}^\star(\rho) \Rightarrow \mathsf{MAC}^\star((m \oplus_2 \langle s, o, \alpha\rangle, f)) \qquad (8)$$

We will say that a state is *secure* if it satisfies each of these three properties. Note that the Bell and La Padula model is only concerned with confidentiality aspects and does not treat integrity and availability of data.

3.3 Requests and answers

The Bell and La Padula model treats three kinds of requests:

1. request by a subject to access to an object in a given mode
2. request by a subject that another subject be given some access attribute with respect to some object
3. request by a subject to create or delete an object from the system

We encode creating or deleting an object as the activation of an unused object index or as its desactivation. This allows to avoid the need to dynamically alter the "dimension of the matrix" (i.e., the fields N_o and N_s of the term $m : \mathcal{M}$ occurring in a state). However, for this created object, we want to be able to set values to its classification and to its needs-to-know: a request altering the classification vector $f : \mathcal{F}$ of a state (with the operator $\oslash$ defined above) will be defined.

Following the notation introduced in the Bell and La Padula model, we define the set $\mathcal{S}^+$ by adding an "empty" element σ_ϵ to $\mathcal{S}$:

Inductive $\mathcal{S}^+$: Set := $\sigma : \mathcal{S} \to \mathcal{S}^+$ | $\sigma_\epsilon : \mathcal{S}^+$.

Hence, an element of $\mathcal{S}^+$ is either $\sigma(s)$, where s belongs to $\mathcal{S}$, or σ_ϵ. The element σ_ϵ is used for requests for which no "target" subject is required.

We define $\mathcal{RA}$ as the set of request attributes (`G` for "get" or "give", `R` for "release" or "rescind", `C` for "change" or "create" and `D` for "delete"):

Inductive $\mathcal{RA}$:Set := G:$\mathcal{RA}$ | R:$\mathcal{RA}$ | C:$\mathcal{RA}$ | D:$\mathcal{RA}$.

We can now define the main parameter of a request: it is either an access attribute, an "empty" element, or the classification level and the needs-to-know associated with an object. So we define the type of the parameter of a request as:

Inductive χ : Set := $\chi_\mathcal{A} : \mathcal{A} \to \chi \mid \chi_\epsilon : \chi \mid \chi_\mathcal{F} : \mathcal{O} \to \mathcal{C} \to [\![\mathcal{K}]\!] \to \chi$.

Note that the constructor $\chi_\mathcal{F}$ will be used for parameters in requests that are intended to set values to the classification and the needs-to-know of an object (when creating or deleting an object for example). Here, we slightly differ from the original paper of Bell and La Padula in which the type of the constructor $\chi_\mathcal{F}$ is $\mathcal{F} \to \chi$. Indeed, instead of changing the classification vector, we just change one of its "entry", in order to limit the enumeration of subjects and objects during the proof. However, this is not an "heavy" restriction since it suffices to consider several requests changing one "entry" to perform several changes in a classification vector.

We are now in position to define the type $\mathcal{R}$ of requests as follows:

Definition $\mathcal{R}$: Set := $\mathcal{S}^+ \times \mathcal{RA} \times \mathcal{S}^+ \times \mathcal{O} \times \chi$.

The possible answers to a request, also called decisions, are `yes` (the request is granted), `no` (the request is not granted) or `undef` (the request is not recognized, no rule is applicable). So, the type $\mathcal{D}$ of answers is defined as follows:

Inductive $\mathcal{D}$: Set := yes:$\mathcal{D}$ | no:$\mathcal{D}$ | undef:$\mathcal{D}$.

In the original paper of Bell and La Padula, a supplementary possible answer, the `error` answer, was considered in order to indicate that the decision-making mechanism was confused. Since such an answer is only used during a debugging step, and then does not appear as an answer in the proposed model, we do not include the `error` answer in our definition.

3.4 Transitions

A request q defines a transition from a state ρ to a state ρ' together with an answer δ. Hence we define transition functions as mappings from $\mathcal{R} \times \Sigma$ to $\mathcal{D} \times \Sigma$:

Definition $\mathcal{T}$: Set := $\mathcal{R} \times \Sigma \to \mathcal{D} \times \Sigma$.

We define a predicate Sound:$\mathcal{T} \to$ `Prop` characterising transition functions that preserve security properties. A transition function that maps every state satisfying the DAC, MAC and MAC* properties, into a state satisfying again these properties is said to be sound. Hence, for a transition fonction $t : \mathcal{T}$, $\mathsf{Sound}(t)$ is defined as the following proposition:

$$
\begin{array}{ll}
\forall q : \mathcal{R}\ \forall \rho : \Sigma & \\
& (\mathsf{DAC}(\rho) \land \mathsf{MAC}(\rho) \land \mathsf{MAC}^\star(\rho)) \\
\Rightarrow & (\mathsf{DAC}(\pi_2(t(q,\rho))) \land \mathsf{MAC}(\pi_2(t(q,\rho))) \land \mathsf{MAC}^\star(\pi_2(t(q,\rho))))
\end{array}
$$

where π_2 stands for the second projection over a pair (i.e., $\pi_2((a,b)) = b$).

It is important to note that this property can be proved for several transition functions and does not capture “basic security properties”. Indeed, in [9], J. McLean defines a transition function that violates “basic security properties” and for which this property holds. In order to illustrate such problems, in our proof, we will explicit what are the parts of the definition of the Bell and La Padula transition function which are used to obtain the proof.

4 Bell and La Padula transition function

The approach presented in the inital paper of Bell and La Padula consists in the definition of the 10 following rules of transition:

(R_1)	get-read	(R_2)	get-append
(R_3)	get-execute	(R_4)	get-write
(R_5)	release-read/write/all/execute	(R_6)	give-read/write/all/execute
(R_7)	rescind-read/write/all/execute	(R_8)	change-f
(R_8)	create-objecct	(R_{10})	delete-object

which are easily encoded into an algorithm. Our implementation is based on several pattern matchings over parameters of a request, thus progressively refining the considered request until it corresponds exactly to one of the 10 rules of Bell and La Padula. For each case, we will explicitly mention the corresponding rule. The first pattern matching over a request $q = (s_1, p, s_2, o, x)$ is concerned with $p : \mathcal{RA}$. Hence, the transition function t_{BLP} is defined as follows:

$$t_{\mathrm{BLP}} : \mathcal{R} \times \Sigma \to \mathcal{D} \times \Sigma := \lambda q : \mathcal{R}.\lambda \rho : \Sigma. \begin{cases} t_{\mathrm{G}}(q,\rho) & \text{when } q = (s_1, \mathtt{G}, s_2, o, x) \\ t_{\mathrm{R}}(q,\rho) & \text{when } q = (s_1, \mathtt{R}, s_2, o, x) \\ t_{\mathrm{C}}(q,\rho) & \text{when } q = (s_1, \mathtt{C}, s_2, o, x) \\ t_{\mathrm{D}}(q,\rho) & \text{when } q = (s_1, \mathtt{D}, s_2, o, x) \end{cases}$$

where t_{G}, t_{R}, t_{C} and t_{D} are defined as transition functions and proved to be sound according to the desired security properties, in the following subsections (the proofs we present are close to the formal proofs obtained and thus does not correspond to the proofs one can find in the original paper of Bell and La Padula). Our main contribution is the formalisation within Coq of the proof of the following theorem:

Theorem 1 (Basic Security Theorem [2]) $\mathsf{Sound}(t_{\mathrm{BLP}})$

This result ensures that the DAC, MAC and $\mathsf{MAC}^\star$ properties are the invariant satisfied by the abstract machine in every reachable state (under the assumption that the initial state also satisfies this invariant).

4.1 G-transition

The transition function t_{G}, defined in table 1, corresponds to 5 of the 10 rules of Bell and La Padula. Let use prove that, given a state $\rho = (m, f)$ (where $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$) satisfying the DAC, MAC and $\mathsf{MAC}^\star$ properties and a request $q = (s_1, \mathtt{G}, s_2, o, x)$, the state $\pi_2(t_{\mathrm{G}}(q,\rho))$ also satisfies the DAC, MAC and $\mathsf{MAC}^\star$ properties.

Cases 1, 3, 4, 6, 7, 9, 10, 12, 14, 15, 16, 17 and 18. In all these cases, we have $\pi_2(t_{\mathrm{G}}(q,\rho)) = \rho$ which statisfies, by hypothesis, the DAC, MAC and $\mathsf{MAC}^\star$ properties. Of course, none of the conditions required in these cases are used in the proof.

$t_{\mathtt{G}}((s_1, \mathtt{G}, s_2, o, x), \rho) :=$ (where $\rho = (m, f)$ and $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$)
when $s_2 = \sigma(s)$
then when $x = \chi_{\mathcal{A}}(\alpha)$
  then when $s_1 = \sigma(s')$
    then if $\alpha \neq \mathtt{c}$
      then if $\alpha \notin m_{(2)}[s', o] \vee \mathtt{c} \notin m_{(2)}[s', o]$
        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 1 (rule } (R_6))}$
        else $(\mathtt{yes}, (m \oplus_2 \langle s', o, \alpha \rangle, f))$ $\boxed{\text{Case 2 (rule } (R_6))}$
      else $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 3 (rule } (R_6))}$
    when $s_1 = \sigma_\epsilon$
    then when $\alpha = \mathtt{r}$
      then if $\mathtt{r} \notin m_{(2)}[s, o] \vee \varphi_1(s) \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_3(s)$
        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 4 (rule } (R_1))}$
        else if $\mathsf{filter}(\Omega(m, s, \{\mathtt{w}, \mathtt{a}\}), \lambda o' : \mathcal{O}.(\varphi_2(o') \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_4(o'))) = \emptyset$
          then $(\mathtt{yes}, (m \oplus_1 \langle s, o, \mathtt{r} \rangle, f))$ $\boxed{\text{Case 5 (rule } (R_1))}$
          else $(\mathtt{no}, \rho)$ $\boxed{\text{Case 6 (rule } (R_1))}$
      when $\alpha = \mathtt{w}$
      then if $\mathtt{w} \notin m_{(2)}[s, o] \vee \varphi_1(s) \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_3(s)$
        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 7 (rule } (R_4))}$
        else if $\mathsf{filter}(\Omega(m, s, \{\mathtt{r}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \preceq \varphi_2(o') \vee \varphi_4(o') \not\subseteq \varphi_4(o))) \cup$
            $\mathsf{filter}(\Omega(m, s, \{\mathtt{a}\}), \lambda o' : \mathcal{O}.(\varphi_2(o') \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_4(o'))) \cup$
            $\mathsf{filter}(\Omega(m, s, \{\mathtt{w}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \neq \varphi_2(o') \vee \varphi_4(o') \not\equiv \varphi_4(o))) = \emptyset$
          then $(\mathtt{yes}, (m \oplus_1 \langle s, o, \mathtt{w} \rangle, f))$ $\boxed{\text{Case 8 (rule } (R_4))}$
          else $(\mathtt{no}, \rho)$ $\boxed{\text{Case 9 (rule } (R_4))}$
      when $\alpha = \mathtt{e}$
      then if $\mathtt{e} \notin m_{(2)}[s, o]$
        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 10 (rule } (R_3))}$
        else $(\mathtt{yes}, (m \oplus_1 \langle s, o, \mathtt{e} \rangle, f))$ $\boxed{\text{Case 11 (rule } (R_3))}$
      when $\alpha = \mathtt{a}$
      then if $\mathtt{a} \notin m_{(2)}[s, o]$
        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 12 (rule } (R_2))}$
        else if $\mathsf{filter}(\Omega(m, s, \{\mathtt{r}, \mathtt{w}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \preceq \varphi_2(o') \vee \varphi_4(o') \not\subseteq \varphi_4(o))) = \emptyset$
          then $(\mathtt{yes}, (m \oplus_1 \langle s, o, \mathtt{a} \rangle, f))$ $\boxed{\text{Case 13 (rule } (R_2))}$
          else $(\mathtt{no}, \rho)$ $\boxed{\text{Case 14 (rule } (R_2))}$
      when $\alpha = \mathtt{c}$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 15 (rule } (R_6))}$
  when $x = \chi_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 16 (rules } (R_1, R_2, R_3, R_4))}$
  when $x = \chi_{\mathcal{F}}(o', c', E)$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 17 (rules } (R_1, R_2, R_3, R_4))}$
when $s_2 = \sigma_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 18 (rules } (R_1, R_2, R_3, R_4))}$

Table 1: G-transition

Case 2. Let us prove that if $\alpha \neq \mathtt{c}$, $\alpha \in m_{(2)}[s', o]$ and $\mathtt{c} \in m_{(2)}[s', o]$, then the state $\rho' = (m \oplus_2 \langle s', o, \alpha\rangle, f)$ is secure. The DAC, MAC and $\mathsf{MAC}^\star$ properties directly follow from (6), (7) and (8).

Case 5. Let us prove that if $\mathtt{r} \in m_{(2)}[s, o]$, $\varphi_1(s) \not\preceq \varphi_2(o)$, $\varphi_4(o) \subseteq \varphi_3(s)$ and:

$$\mathsf{filter}(\Omega(m, s, \{\mathtt{w},\mathtt{a}\}), \lambda o' : \mathcal{O}.(\varphi_2(o') \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_4(o'))) = \emptyset \tag{9}$$

then the state $\rho' = (m \oplus_1 \langle s, o, \mathtt{r}\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ holds since, by hypothesis, $\mathsf{DAC}(\rho)$ holds and $\mathtt{r} \in m_{(2)}[s, o]$.
Since $\preceq$ is a total order, we have $\varphi_2(o) \preceq \varphi_1(s)$. Thus, since by hypothesis $\varphi_4(o) \subseteq \varphi_3(s)$, $\mathsf{MAC}(\rho')$ holds.
In order to prove $\mathsf{MAC}^\star(\rho')$, let s_0 be a subject and o_1 and o_2 be two objects such that:

$$o_1 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{w},\mathtt{a}\}) \quad o_2 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{r},\mathtt{w}\})$$

Since, $\mathtt{r} \notin \{\mathtt{w},\mathtt{a}\}$, by (2), we have $o_1 \in \Omega(m, s_0, \{\mathtt{w},\mathtt{a}\})$, and because $\mathsf{MAC}^\star(\rho)$ holds, it follows:

$$\forall o' : \mathcal{O}\ o' \in \Omega(m, s_0, \{\mathtt{r},\mathtt{w}\}) \Rightarrow \varphi_2(o') \preceq \varphi_2(o_1) \wedge \varphi_4(o') \subseteq \varphi_4(o_1) \tag{10}$$

Two cases are possible.

1. either $s_0 = s$ and
   - either $o = o_2$ and, by (9), since $o_1 \in \Omega(m, s_0, \{\mathtt{w},\mathtt{a}\})$, we have $\varphi_2(o_1) \not\preceq \varphi_2(o_2)$ – from which we obtain $\varphi_2(o_2) \preceq \varphi_2(o_1)$ because $\preceq$ is a total order – and $\varphi_4(o_2) \subseteq \varphi_4(o_1)$, and we can conclude.
   - or $o \neq o_2$ and, by (3), we have $o_2 \in \Omega(m, s_0, \{\mathtt{r},\mathtt{w}\})$, and, by (10), we can conclude.
2. or $s_0 \neq s$ and, by (2), it follows $o_2 \in \Omega(m, s_0, \{\mathtt{r},\mathtt{w}\})$, and, by (10), we can conclude.

Case 8. Let us prove that if $\mathtt{w} \in m_{(2)}[s, o]$, $\varphi_1(s) \not\preceq \varphi_2(o)$, $\varphi_4(o) \subseteq \varphi_3(s)$ and:

$$\begin{array}{rl} & \mathsf{filter}(\Omega(m, s, \{\mathtt{r}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \preceq \varphi_2(o') \vee \varphi_4(o') \not\subseteq \varphi_4(o))) \\ \cup & \mathsf{filter}(\Omega(m, s, \{\mathtt{a}\}), \lambda o' : \mathcal{O}.(\varphi_2(o') \preceq \varphi_2(o) \vee \varphi_4(o) \not\subseteq \varphi_4(o'))) \\ \cup & \mathsf{filter}(\Omega(m, s, \{\mathtt{w}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \neq \varphi_2(o') \vee \varphi_4(o') \not\equiv \varphi_4(o))) = \emptyset \end{array} \tag{11}$$

then the state $\rho' = (m \oplus_1 \langle s, o, \mathtt{w}\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ holds since, by hypothesis, $\mathsf{DAC}(\rho)$ holds and $\mathtt{w} \in m_{(2)}[s, o]$.
Since $\preceq$ is a total order, we have $\varphi_2(o) \preceq \varphi_1(s)$. Thus, since by hypothesis $\varphi_4(o) \subseteq \varphi_3(s)$, $\mathsf{MAC}(\rho')$ holds.
In order to prove $\mathsf{MAC}^\star(\rho')$, let s_0 be a subject and o_1 and o_2 be two objects such that:

$$o_1 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{w},\mathtt{a}\}) \quad o_2 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{r},\mathtt{w}\})$$

Two cases are possible.

1. either $s_0 = s$ and:
   - either $o = o_1$ and we can distinguish two subcases: if $o_1 = o_2$, then we can conclude since $\preceq$ and $\subseteq$ are reflexive relations, else, we have $o_2 \neq o$, and, by (3), from $o_2 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{r},\mathtt{w}\})$, it follows $o_2 \in \Omega(m, s_0, \{\mathtt{r},\mathtt{w}\})$, so, by (1), either $o_2 \in \Omega(m, s_0, \{\mathtt{r}\})$, and, by (11), we can conclude since $\preceq$ is a total order, or $o_2 \in \Omega(m, s_0, \{\mathtt{w}\})$, and now, from (11), it follows $\varphi_2(o_1) = \varphi_2(o_2)$ and $\varphi_4(o_1) \equiv \varphi_4(o_2)$ and we can conclude since $\preceq$ and $\subseteq$ are reflexive relations.

- or $o \neq o_1$ and, by (3), $o_1 \in \Omega(m, s_0, \{\mathtt{w,a}\})$ and we can distinguish two subcases: if $o_2 \neq o$, then, by (3), we have $o_2 \in \Omega(m, s_0, \{\mathtt{r,w}\})$, and we can conclude, since, by hypothesis, $\mathsf{MAC}^\star(\rho)$ holds, or $o_2 = o$ and then, from $o_1 \in \Omega(m, s_0, \{\mathtt{w,a}\})$, by (1), either $o_1 \in \Omega(m, s_0, \{\mathtt{w}\})$ and, by (11), we obtain $\varphi_2(o_1) = \varphi_2(o_2)$ and $\varphi_4(o_1) \equiv \varphi_4(o_2)$ which allows to conclude (since $\preceq$ and $\subseteq$ are reflexive relations) or $o_1 \in \Omega(m, s_0, \{\mathtt{a}\})$ and, by (11), we can conclude since $\preceq$ is a total order

2. or $s_0 \neq s$ and then, by (2), we have:

$$\begin{aligned}\Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{w,a}\}) &= \Omega(m, s_0, \{\mathtt{w,a}\})\\ \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{r,w}\}) &= \Omega(m, s_0, \{\mathtt{r,w}\})\end{aligned}$$

and the conclusion follows directly from the hypothesis $\mathsf{MAC}^\star(\rho)$.

Case 11. Let us prove that if $\mathtt{e} \in m_{(2)}[s, o]$, then the state $\rho' = (m \oplus_1 \langle s, o, \mathtt{e}\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ holds since, by hypothesis, $\mathsf{DAC}(\rho)$ holds and $\mathtt{e} \in m_{(2)}[s, o]$.
$\mathsf{MAC}(\rho')$ holds since, by hypothesis, $\mathsf{MAC}(\rho)$ holds and is only concerned with the access attributes $\mathtt{r}$ and $\mathtt{w}$.
$\mathsf{MAC}^\star(\rho')$ holds since, by hypothesis, $\mathsf{MAC}^\star(\rho)$ holds and, since $\mathtt{e} \notin \{\mathtt{w,a}\}$ and $\mathtt{e} \notin \{\mathtt{r,w}\}$, by (2), we have:

$$\begin{aligned}\Omega(m \oplus_1 \langle s, o, \mathtt{e}\rangle, s_0, \{\mathtt{w,a}\}) &= \Omega(m, s_0, \{\mathtt{w,a}\})\\ \Omega(m \oplus_1 \langle s, o, \mathtt{e}\rangle, s_0, \{\mathtt{r,w}\}) &= \Omega(m, s_0, \{\mathtt{r,w}\})\end{aligned}$$

Case 13. Let us prove that if $\mathtt{a} \in m_{(2)}[s, o]$ and

$$\mathsf{filter}(\Omega(m, s, \{\mathtt{r,w}\}), \lambda o' : \mathcal{O}.(\varphi_2(o) \preceq \varphi_2(o') \vee \varphi_4(o') \not\subseteq \varphi_4(o))) = \emptyset \tag{12}$$

then the state $\rho' = (m \oplus_1 \langle s, o, \mathtt{a}\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ holds since, by hypothesis, $\mathsf{DAC}(\rho)$ holds and $\mathtt{a} \in m_{(2)}[s, o]$.
$\mathsf{MAC}(\rho')$ holds since, by hypothesis, $\mathsf{MAC}(\rho)$ holds and is only concerned with the access attribute $\mathtt{r}$ and $\mathtt{w}$.
In order to prove $\mathsf{MAC}^\star(\rho')$, let s_0 be a subject and o_1 and o_2 be two objects such that:

$$o_1 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{w,a}\}) \quad o_2 \in \Omega(m \oplus_1 \langle s, o, r\rangle, s_0, \{\mathtt{r,w}\})$$

Since $\mathtt{a} \notin \{\mathtt{r,w}\}$, by (2), we have $o_2 \in \Omega(m, s_0, \{\mathtt{r,w}\})$. Now, if $s_0 \neq s$, then, by (2), we also have $o_1 \in \Omega(m, s_0, \{\mathtt{w,a}\})$ and $\mathsf{MAC}^\star(\rho')$ holds since, by hypothesis, $\mathsf{MAC}^\star(\rho)$ holds, else, if $s_0 = s$, then either $o_1 \neq o$, and, by (3), we have $o_1 \in \Omega(m, s_0, \{\mathtt{w,a}\})$ and $\mathsf{MAC}^\star(\rho')$ holds again by hypothesis, or $o_1 = o$ and then from $s_0 = s$ we obtain $o_2 \in \Omega(m, s, \{\mathtt{r,w}\})$ and, by (12), we have $\varphi_2(o_1) \not\preceq \varphi_2(o_2)$ and $\varphi_4(o_2) \subseteq \varphi_4(o_1)$, and because $\preceq$ is a total order it follows $\varphi_2(o_2) \preceq \varphi_2(o_1)$ and we can conclude.

4.2 R-transition

The transition function t_{R} is defined in table 2. Let use prove that, given a state $\rho = (m, f)$ (where $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$) satisfying the DAC, MAC and $\mathsf{MAC}^\star$ properties and a request $q = (s_1, \mathtt{R}, s_2, o, x)$, the state $\pi_2(t_{\mathrm{R}}(q, \rho))$ also satisfies the DAC, MAC and $\mathsf{MAC}^\star$ properties.

Cases 1, 2, 3, 4 and 6 In all these cases, we have $\pi_2(t_{\mathrm{R}}(q, \rho)) = \rho$ which statisfies, by hypothesis, the DAC, MAC and $\mathsf{MAC}^\star$ properties. Of course, none of the conditions required in these cases are used in the proof.

$t_{\mathrm{R}}((s_1, \mathtt{R}, s_2, o, x), \rho) :=$ (where $\rho = (m, f)$ and $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$)

when $x = \chi_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 1 (rules } (R_5, R_7))}$

when $x = \chi_{\mathcal{F}}(o', c', E)$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 2 (rules } (R_5, R_7))}$

when $x = \chi_{\mathcal{A}}(\alpha)$

then when $s_2 = \sigma_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 3 (rules } (R_5, R_7))}$

  when $s_2 = \sigma(s)$

  then if $\alpha \notin \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{e}\}$

    then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 4 (rules } (R_5, R_7))}$

    else when $s_1 = \sigma_\epsilon$

      then $(\mathtt{yes}, (m \ominus_1 \langle s, o, \alpha \rangle, f))$ $\boxed{\text{Case 5 (rule } (R_5))}$

      when $s_1 = \sigma(s')$

      then if $\alpha \notin m_{(2)}[s', o] \vee \mathtt{c} \notin\in m_{(2)}[s', o]$

        then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 6 (rule } (R_7))}$

        else $(\mathtt{yes}, ((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle, f))$ $\boxed{\text{Case 7 (rule } (R_7))}$

Table 2: R-transition

Case 5 Let us prove that if $x = \chi_{\mathcal{A}}(\alpha)$ with $\alpha \in \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{e}\}$, $s_2 = \sigma(s)$ and $s_1 = \sigma_\epsilon$, then the state $\rho' = (m \ominus_1 \langle s, o, \alpha \rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ directly follows from (4) since, by hypothesis, $\mathsf{DAC}(\rho)$ holds.
$\mathsf{MAC}(\rho')$ directly follows from (4) since, by hypothesis, $\mathsf{MAC}(\rho)$ holds.
$\mathsf{MAC}^\star(\rho')$ directly follows from (5) since, by hypothesis, $\mathsf{MAC}^\star(\rho)$ holds.

Note that we prove these properties without using the hypothesis $\alpha \in \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{e}\}$ and $s_1 = \sigma_\epsilon$.

Case 7 Let us prove that if $x = \chi_{\mathcal{A}}(\alpha)$ with $\alpha \in \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{e}\}$, $s_2 = \sigma(s)$, $s_1 = \sigma(s')$, and $\alpha \in m_{(2)}[s', o]$, and $\mathtt{c} \in m_{(2)}[s', o]$ then the state $\rho' = ((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle, f)$ is secure.
In order to prove $\mathsf{DAC}(\rho')$, let s_0 be a subject and o_0 be an object. If

$$\alpha_0 \in ((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$$

then, by definition of $\ominus_2$, we have $\alpha_0 \in (m \ominus_1 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$. Two cases are possible:

1. either $s \neq s_0 \vee o_0 \neq o$ and it follows $\alpha_0 \in m_{(1)}[s_0, o_0]$, so, since, by hypothesis, $\mathsf{DAC}(\rho)$ holds, we obtain $\alpha_0 \in m_{(2)}[s_0, o_0]$, and, by definition of $\ominus_1$, we have $\alpha_0 \in (m \ominus_1 \langle s, o, \alpha \rangle)_{(2)}[s_0, o_0]$, and then we obtain $\alpha_0 \in ((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$ since $s \neq s_0 \vee o_0 \neq o$.
2. or $s = s_0$ and $o_0 = o$ and, because $\alpha_0 \in (m \ominus_1 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$, we have $\alpha \neq \alpha_0$, and it follows $\alpha_0 \in m_{(1)}[s_0, o_0]$, so, since, by hypothesis, $\mathsf{DAC}(\rho)$ holds, we obtain $\alpha_0 \in m_{(2)}[s_0, o_0]$, and, by definition of $\ominus_1$, we have $\alpha_0 \in (m \ominus_1 \langle s, o, \alpha \rangle)_{(2)}[s_0, o_0]$, and then, since $\alpha \neq \alpha_0$, we obtain $\alpha_0 \in ((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$.

By definition of $\ominus_2$, forall subject s_0 and forall object o_0, we have:

$$((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0] = (m \ominus_1 \langle s, o, \alpha \rangle)_{(1)}[s_0, o_0]$$

and $\mathsf{MAC}(\rho')$ directly follows from (4) since, by hypothesis, $\mathsf{MAC}(\rho)$ holds.

$t_{\mathrm{C}}((s_1, \mathtt{C}, s_2, o, x), \rho) :=$ (where $\rho = (m, f)$ and $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$)

when $s_1 = \sigma(s)$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 1 (rules } (R_8, R_9))}$

when $s_1 = \sigma_\epsilon$

then when $s_2 = \sigma_\epsilon$

  then when $x = \chi_{\mathcal{A}}(\alpha)$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 2 (rule } (R_8))}$

    when $x = \chi_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 3 (rule } (R_8))}$

    when $x = \chi_{\mathcal{F}}(o', c', E)$

    then if $o' \circledin m$

      then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 4 (rule } (R_8))}$

      else $(\mathtt{yes}, (m, f \oslash \langle o', c', E \rangle))$ $\boxed{\text{Case 5 (rule } (R_8))}$

  when $s_2 = \sigma(s')$

  then if $x = \chi_{\mathcal{A}}(\mathtt{e}) \vee x = \chi_\epsilon$

    then if $o \circledin m$

      then $(\mathtt{no}, \rho)$ $\boxed{\text{Case 6 (rule } (R_9))}$

      else when $x = \chi_\epsilon$

        then $(\mathtt{yes}, (m \circleddash \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\}) \rangle, f))$ $\boxed{\text{Case 7 (rule } (R_9))}$

        when $x = \chi_{\mathcal{A}}(\alpha)$

        then $(\mathtt{yes}, (m \circleddash \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}, \mathtt{e}\}) \rangle, f))$ $\boxed{\text{Case 8 (rule } (R_9))}$

        when $x = \chi_{\mathcal{F}}(o', c', E)$

        then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 9 (rule } (R_9))}$

    else $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 10 (rule } (R_9))}$

Table 3: C-transition

Here again, we can prove that forall subject s_0, forall (finite) set E of access attributes, we have:

$$\Omega(((m \ominus_1 \langle s, o, \alpha \rangle) \ominus_2 \langle s, o, \alpha \rangle), s_0, E) = \Omega(m \ominus_1 \langle s, o, \alpha \rangle), s_0, E)$$

and $\mathsf{MAC}^\star(\rho')$ directly follows from (5) since, by hypothesis, $\mathsf{MAC}^\star(\rho)$ holds.

Note that we prove these properties without using the hypothesis $\alpha \in \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{e}\}$, $s_1 = \sigma(s')$, $\alpha \in m_{(2)}[s', o]$, and $\mathtt{c} \in m_{(2)}[s', o]$.

4.3 C-transition

The transition function t_{C} is defined in table 3. Let use prove that, given a state $\rho = (m, f)$ (where $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$) satisfying the DAC, MAC and $\mathsf{MAC}^\star$ properties and a request $q = (s_1, \mathtt{C}, s_2, o, x)$, the state $\pi_2(t_{\mathrm{C}}(q, \rho))$ also satisfies the DAC, MAC and $\mathsf{MAC}^\star$ properties.

Cases 1, 2, 3, 4, 6, 9 and 10 In all these cases, we have $\pi_2(t_{\mathrm{C}}(q, \rho)) = \rho$ which statisfies, by hypothesis, the DAC, MAC and $\mathsf{MAC}^\star$ properties. Of course, none of the conditions required in these cases are used in the proof.

Case 5 Let us prove that if $s_1 = s_2 = \sigma_\epsilon$, $x = \chi_{\mathcal{F}}(o', c', E)$ and $o' \not\circledin m$ then the state $\rho' = (m, f \oslash \langle o', c', E \rangle)$ is secure.

The DAC property is only concerned with the matrix m which is not modified. Thus, $\mathsf{DAC}(\rho')$ directly follows from the hypothesis $\mathsf{DAC}(\rho)$.
$\mathsf{MAC}(\rho')$ holds since given a subject $s_0 : \mathcal{S}$ and an object $o_0 : \mathcal{O}$, either $o_0 \neq o'$ and we can conclude since $\mathsf{MAC}(\rho)$ holds, or $o_0 = o'$ and then, since, by hypothesis, $o' \not\circledcirc m$, we have $m_{(2)}[s_0, o'] = \emptyset$, and because, by hypothesis, $\mathsf{DAC}(\rho)$ holds, it follows $m_{(1)}[s_0, o'] = \emptyset$ and we can conclude since neither $\mathtt{r}$ nor $\mathtt{w}$ can belong to $m_{(1)}[s_0, o']$.
In order to prove $\mathsf{MAC}^\star(\rho')$, let $s_0 : \mathcal{S}$ be a subject and o_1 and o_2 be two objects such that $o_1 \in \Omega(m, s_0, \{\mathtt{w}, \mathtt{a}\})$ and $o_2 \in \Omega(m, s_0, \{\mathtt{r}, \mathtt{w}\})$. Here again, from the hypothesis $o' \not\circledcirc m$, we have $m_{(2)}[s_0, o'] = \emptyset$, and because, by hypothesis, $\mathsf{DAC}(\rho)$ holds, it follows $m_{(1)}[s_0, o'] = \emptyset$, so $o_1 \neq o'$ and $o_2 \neq o'$, and we obtain $\mathsf{MAC}^\star(\rho')$ as a direct consequence of the hypothesis $\mathsf{MAC}^\star(\rho)$.

Case 7 Let us prove that if $s_1 = \sigma_\epsilon$, $s_2 = \sigma(s')$, $x = \chi_\epsilon$ and $o \not\circledcirc m$ then the state $\rho' = (m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ follows from $\mathsf{DAC}(\rho)$ and:

$$\begin{array}{rl} & (m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle)_{(1)}[s', o] = \emptyset \\ \subseteq & (m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle)_{(2)}[s', o] = \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\} \end{array}$$

$\mathsf{MAC}(\rho')$ holds since $(m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle)_{(1)}[s', o] = \emptyset$ and then neither $\mathtt{r}$ nor $\mathtt{w}$ can belong to this (empty) set. Hence the hypothesis $\mathsf{MAC}(\rho)$ allows to conclude.
In order to prove $\mathsf{MAC}^\star(\rho')$, let $s_0 : \mathcal{S}$ be a subject and o_1 and o_2 be two objects such that:

$$\begin{array}{l} o_1 \in \Omega(m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle, s_0, \{\mathtt{w}, \mathtt{a}\}) \\ o_2 \in \Omega(m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle, s_0, \{\mathtt{r}, \mathtt{w}\}) \end{array}$$

Two cases are possible. Either $s' = s_0$ and then, here again, since:

$$(m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle)_{(1)}[s', o] = \emptyset$$

we have $o_1 \neq o$ and $o_2 \neq o$, and we obtain $\mathsf{MAC}^\star(\rho')$ as a direct consequence of the hypothesis $\mathsf{MAC}^\star(\rho)$, or $s' \neq s_0$ and we have:

$$(m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}\})\rangle)_{(1)}[s', o] = m_{(1)}[s_0, o]$$

and the hypothesis $\mathsf{MAC}^\star(\rho)$ allows again to conclude.

Note that we prove these properties without using the hypothesis $o \not\circledcirc m$.

Case 8 If $s_1 = \sigma_\epsilon$, $s_2 = \sigma(s')$, $x = \chi_{\mathcal{A}}(\alpha)$ and $o \not\circledcirc m$ then the proof that the state $\rho' = (m \ominus \langle s', o, (\emptyset, \{\mathtt{r}, \mathtt{w}, \mathtt{a}, \mathtt{c}, \mathtt{e}\})\rangle, f)$ is secure is exactly the same as for the case 7.

4.4 D-transition

The transition function t_{D} is defined in table 4. Let use prove that, given a state $\rho = (m, f)$ (where $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$) satisfying the DAC, MAC and $\mathsf{MAC}^\star$ properties and a request $q = (s_1, \mathtt{D}, s_2, o, x)$, the state $\pi_2(t_{\mathrm{D}}(q, \rho))$ also satisfies the DAC, MAC and $\mathsf{MAC}^\star$ properties.

Cases 1, 3 and 4 In all these cases, we have $\pi_2(t_{\mathrm{D}}(q, \rho)) = \rho$ which statisfies, by hypothesis, the DAC, MAC and $\mathsf{MAC}^\star$ properties. Of course, none of the conditions required in these cases are used in the proof.

$t_{\mathtt{D}}((s_1, \mathtt{D}, s_2, o, x), \rho) :=$ (where $\rho = (m, f)$ and $f = (\varphi_1, \varphi_2, \varphi_3, \varphi_4)$)
when $s_2 = \sigma(s)$
then if $s_1 \neq \sigma_\epsilon \vee x \neq \chi_\epsilon$
  then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 1 (rule } (R_{10}))}$
  else if $\mathtt{c} \in m_{(2)}[s, o]$
    then $(\mathtt{yes}, (m \ominus \langle s, o, (\emptyset, \emptyset)\rangle, f))$ $\boxed{\text{Case 2 (rule } (R_{10}))}$
    else $(\mathtt{no}, \rho)$ $\boxed{\text{Case 3 (rule } (R_{10}))}$
when $s_2 = \sigma_\epsilon$ then $(\mathtt{undef}, \rho)$ $\boxed{\text{Case 4 (rule } (R_{10}))}$

Table 4: D-transition

Case 2 Let us prove that if $s_2 = \sigma(s)$, $s_1 = \sigma_\epsilon$, $x = \chi_\epsilon$ and $\mathtt{c} \in m_{(2)}[s, o]$ then the state $\rho' = (m \ominus \langle s, o, (\emptyset, \emptyset)\rangle, f)$ is secure.
$\mathsf{DAC}(\rho')$ follows from $\mathsf{DAC}(\rho)$ and:

$$(m \ominus \langle s, o, (\emptyset, \emptyset)\rangle)_{(1)}[s, o] = \emptyset \subseteq (m \ominus \langle s, o, (\emptyset, \emptyset)\rangle)_{(2)}[s, o] = \emptyset$$

$\mathsf{MAC}(\rho')$ holds since, by hypothesis, we have $\mathsf{MAC}(\rho)$ and furthermore neither $\mathtt{r}$ nor $\mathtt{w}$ can belong to $(m \ominus \langle s, o, (\emptyset, \emptyset)\rangle)_{(1)}[s, o] = \emptyset$.
$\mathsf{MAC}^\star(\rho')$ holds since, by hypothesis, we have $\mathsf{MAC}^\star(\rho)$ and any object o' belonging either to $\Omega(m \ominus \langle s, o, (\emptyset, \emptyset)\rangle, s_0, \{\mathtt{w}, \mathtt{a}\})$ or to $\Omega(m \ominus \langle s, o, (\emptyset, \emptyset)\rangle, s_0, \{\mathtt{r}, \mathtt{w}\})$ is necessarily such that $o' \neq o$.

Note that we prove these properties without using the hypothesis $\mathtt{c} \in m_{(2)}[s, o]$.

5 Conclusion – Future work

This development shows that even from a pratical point of view, formal methods can be used to increase the security of a software, by providing a mathematical model and using it to prove the desired security properties. In fact, using mathematical concepts does not always produce very complex descriptions of systems and the proofs we have formalised here are not very difficult to obtain within the Coq proof assistant. Furthermore, by providing a way to extract programs from formal developments, the Coq proof assistant avoids to treat separately the formal modeling work and the development.

Of course, the proofs presented in this paper are now well-known and one can wonder about the usefulness of such a formalisation. However, formalising proofs brings us at a level of detail that is often left to the reader, and by taking into account these details, often considered as minor in informal presentations, proofs are getting a little more complicated. Furthermore, our development formally ensures that the program we have obtained satisfies the desired security properties, thus providing a greater level of confidence (while, in a paper, a typographical error can be seen as a small mistake, in an implementation, it becomes a bug that can cause serious damages).

The relevance of the choice of the Bell and La Padulla model can also be addressed. Indeed, this model is not completely satisfactory. For example, as we said, it is only concerned with confidentiality. Furthermore, the "basic security theorem" proved (theorem 1) does not capture the "essence" of security. Indeed, as we said, in [9], J. McLean proves an essentially similar theorem for a model that clearly violates basic notions of security. Such a work is cited in [16] by J. Rushby which presents a more general discussion about problems on specifying security requirements. In [11], J. McLean addresses this problem by defining a framework of security models that contains transition restrictions. Roughly speaking, a function from $\mathcal{S} \cup \mathcal{O}$ to $[\![\mathcal{S}]\!]$ which returns the set of subjects that are allowed to change the security level of its argument is introduced.

However, the Bell and La Padula model has been widely used as a basis for designing systems with specified security properties. For example, in [4, 5], P. Bieber presents the development of a secure gateway, called FOX, that interconnects two local area networks and allowing a security evaluation at ITSEC [6] assurance-correctness level E4, requiring the use of formal methods. The security policy of FOX, concerned with two kinds of properties: information isolation and information filtration (between the two LAN), has been enforced by a formal security policy model based on the access control rules of the MULTICS interpretation of the Bell and La Padulla model [3]. For this, the consistency of this model has been checked using tools associated with the B-method [1] and last, this formal model has been interpreted in order to show that security is correctly taken into account in the specification of FOX. Each functional requirement has been related with access control rules of the model, thus showing that the functional specification of FOX is consistent with its security model.

Despite of these remarks, we think that this development can be viewed as a base for several extensions. For example, formalising the framework presented in [11] or adding several levels of refinements (together with a formal proof of the coherence of such extensions) could be interesting future works.

References

[1] J.R. Abrial. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, 1996.

[2] D. Bell and L. LaPadula. Secure Computer Systems: a Mathematical Model. Technical Report MTR-2547 (Vol. II), MITRE Corp., Bedford, MA, May 1973. Reprinted in [8].

[3] D. Bell and L. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR-2997, MITRE Corp., Bedford, MA, July 1975.

[4] P. Bieber. Formal Techniques for an ITSEC-E4 Secure Gateway. In *12th Annual Computer Security Applications Conference*, pages 236–245, December 1996.

[5] P. Bieber. Interprétation d'un modèle de sécurité. *Techniques et Sciences Informatiques*, 15(6), 1996.

[6] European Economic Communuty. Information Technology Security Evaluation Criteria (ITSEC). Technical report, CEE, 1990.

[7] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2-3):95–120, 1988.

[8] L.J. LaPadula and D.E. Bell. Secure Computer Systems: A Mathematical Model. *Journal of Computer Security*, 4:239–263, 1996.

[9] J. McLean. A comment on the 'basic security theorem' of Bell and LaPadula. *Information Processing Letters*, 20(2):67–70, February 1985.

[10] J. McLean. Security models and information flow. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 180–187, 1990.

[11] J. McLean. The specifications and modeling of computer security. *IEEE Computer*, 23(1):9–16, 1990.

[12] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. J. Wiley & Sons, 1994.

[13] NSA/Information Assurance Directorate. Protection Profile For Multilevel Operating Systems In Environments Requiring Medium Robustness. Technical report, National Security Agency, 23 May 2001.

[14] C. Paulin-Mohring and B. Werner. Synthesis of ML programs in the system Coq. *Journal of Symbolic Computation*, 15(5–6):607–640, 1993.

[15] Logical Project. *The Coq Proof Assistant Reference Manual Version 7*. INRIA-Rocquencourt, 2002.

[16] J. Rushby. Security requirements specifications: How and what? In *Symposium on Requirements Engineering for Information Security (SREIS)*, Indianapolis, IN, March 2001.