In CONCUR 2001, Lecture Notes in Computer Science, 2154:102–120, © Springer, 2001.

Reasoning about Security in Mobile Ambients ⋆

Michele Bugliesi[1], Giuseppe Castagna[2], and Silvia Crafa[1,2]

[1] Dipartimento di Informatica
Univ. "Ca' Foscari", Venezia, Italy

[2] Département d'Informatique
École Normale Supérieure, Paris, France

Abstract. The paper gives an assessment of security for *Mobile Ambients*, with specific focus on *mandatory access control* (MAC) policies in multilevel security systems. The first part of the paper reports on different formalization attempts for MAC policies in the Ambient Calculus, and provides an in-depth analysis of the problems one encounters. As it turns out, MAC security does not appear to have fully convincing interpretations in the calculus. The second part proposes a solution to this *impasse*, based on a variant of Mobile Ambients. A type system for resource access control is defined, and the new calculus is discussed and illustrated with several examples of resource management policies.

1 Introduction

Distributed computation based on mobile code is already ubiquitous and represents an essential aspect of our computing environments. Mobile computing relies on sharing of data and software resources among computing sites distributed across wide-area open networks. This sharing is successful inasmuch as it satisfies several criteria, including safety, e.g. execution of mobile code without failure, and security, e.g. protection of sites against malicious intruders and misuse of their computing resources.

A substantial body of the research on programming languages has recently been directed towards the study of formal calculi providing high-level support for mobile agents. A non exhaustive list of examples includes the Ambient Calculus [CG98], the Seal Calculus [VC99,CGN01], the Dπ-calculus [HR00b], and the Join Calculus [FGL$^+$96].

The initial motivation for this paper was an assessment of security in calculi for mobility. As a preliminary step, we thought it instructive to study what (if any) new insight and challenges mobile code languages provide for well-established security models. For some of the calculi we just mentioned, notably for the Dπ-calculus, an in-depth study of these aspects has already been conducted in [HR00b]. Here we present a corresponding analysis for Mobile Ambients, for which, to our knowledge, no previous attempt in this direction has been made.

The focus of our analysis is on *mandatory access control* policies (MAC) in multilevel security systems. In particular, the emphasis is on the specific aspects of MAC policies related to confidentiality and integrity, and their different implementations as *military* security (no read-up, no write-down) and *commercial* security (no read-up, no write-up).

The first part of the paper (§ 2) is a survey of our formalization attempts. As it turns out, the main problem comes far ahead the point where one starts the formalization, because the security concepts assumed as references do not appear to have any fully convincing interpretation in the calculus. In fact, the very meaning of basis notions such as "read access" and "write access" by subjects on objects, or even "ownership", is somehow difficult to grasp and characterize when looked at from within the Ambient Calculus. As a consequence of these difficulties, one is led to the conclusion that Ambients lack adequate primitives to capture and characterize those security concepts. While our arguments are only informal, the analysis we detail in the first part of the paper does provide convincing evidence in favor our conclusion.

⋆ Work partially supported by MURST Project 9901403824_003, by CNRS Program *Telecommunications*: "Collaborative, distributed, and secure programming for Internet", and by Galileo Action n. 02841UD

The second part of the paper proposes a solution to this *impasse*, based on a variant of Mobile Ambients we dub *Boxed Ambients*. The calculus of Boxed Ambients is introduced, formally defined and studied in a companion paper [BCC01]. Here, instead, we keep the presentation largely informal, and put the emphasis on the role of the new calculus for describing and expressing resource access control policies. After a brief description of the calculus, we introduce a type system for resource access control (§ 3). Then (§ 4) we propose several examples that illustrate what we believe to be the merits and strengths of the calculus. A final section (§ 5) is dedicated to related work and conclusions.

While the second part of the paper represents the main contribution of the paper, the preliminary analysis was extremely useful to us to understand the problems, and we hope will be equally valuable to the reader.

2 Mobile Ambients and Multilevel Security

Standard models of security for resource access control are built around *subjects* performing access requests on *objects* by *write* (in some models, also *append*, *execute*, and others) and *read* operations.

Multilevel security presupposes a lattice of security levels, and every subject and object is assigned a level in this lattice. Based on these levels, access to objects by subjects are classified as *read-up* (resp. *read-down*) when a subject access by a read an object of higher (resp. lower) level, and similarly for write accesses. Relying on this classification, one may distinguish two security policies: *military* security, which forbids (both direct and indirect) read-up's and write-down's, and *commercial* security that forbids read-up's and write-up's. These notions cover also *indirect* accesses resulting from the composition of atomic operations: thus also the fact of writing into an object of any level a piece of information read (or just coming) from another object whose level is higher than the level of the first object is considered as a write-down (classic security handles these cases by the so-called $\star$-property [BP76,Gol99][1]).

2.1 Mobile Ambients

Ambients are processes of the form $a[\![\, P \,]\!]$ where a is a name and P a process. Processes can be composed in parallel, as in $P \mid Q$, exercise a capability, as in $M.P$, declare local names as in $(\boldsymbol{\nu} x)P$, they can be replicated, as in $!P$, or simply do nothing as in $\mathbf{0}$.

Mobility. Ambients may be nested to form a tree structure that can be dynamically reconfigured as a result of mobility and ambient dissolution determined by the capabilities in, out and open. To exemplify, consider the ambients a and b in the configuration $a[\![\, \mathsf{open}\ b.\mathsf{in}\ c \,]\!] \mid b[\![\, \mathsf{in}\ a.\mathsf{in}\ d \,]\!]$. The ambient b may enter a, by exercising the *capability* $\mathsf{in}\ a$, and reduce to $a[\![\, \mathsf{open}\ b.\mathsf{in}\ c \mid b[\![\, \mathsf{in}\ d \,]\!] \,]\!]$. Then a may dissolve b by exercising $\mathsf{open}\ b$, and reduce to $a[\![\, \mathsf{in}\ c \mid \mathsf{in}\ d \,]\!]$.

Security. The ability or inability to cross boundaries, which is conferred by the capabilities in and out, is also at the core of the security model underlying Mobile Ambients. Permission to cross ambient boundaries is given by making the name available to the clients willing to enter or exit. Names are thus viewed as passwords, or alternatively as cryptokeys: when embedded in a capability, an ambient name provides the pass that enables access to, or else the cryptokey that discloses the contents of that ambient.

While this model of security is suggestive, and powerful for its simplicity, it appears to not be fully adequate for modeling realistic policies for resource access control. The problem is that it entirely depends on the ability by the authorization mechanism to filter out undesired clients: an

[1] As a matter of fact, these references do not define precisely what a *write-down access* is; instead, they give a definition of *no-write down policy*.

authorization breach could grant malicious agents full access to all the resources located inside the ambient boundary. Clearly, one first has to identify what "resource access" is in the Ambient Calculus. Entering an ambient, or opening it are all good notions of access: in addition, there is of course communication.

Communication. In the Ambient Calculus, communication is anonymous, and happens inside ambients. The configuration $(x)P \mid \langle M \rangle$ represents the parallel composition of two processes, the output process $\langle M \rangle$ "dropping" the message M, and the input process $(x)P$ reading the message M and continuing as $P\{x := M\}$. The open capability has a fundamental interplay with this form of communication: opening an ambient enables synchronization between the processes located in the opening and the opened ambients. To exemplify, synchronization between the input process $(x)P$ and the output $\langle M \rangle$ in the system $(x)P \mid \mathsf{open}\ b \mid b[\![\langle M \rangle \mid Q]\!]$ is enabled by exercising the capability $\mathsf{open}\ b$ to unleash the message $\langle M \rangle$.

It is the interplay between communication and the primitives for ambient mobility which makes it difficult to reason about resource access in terms of classical security models. To make our point, we use a simple concrete example.

2.2 A simple resource access problem

Suppose we have a system consisting of a set of resources $\{r_1, \ldots, r_n\}$ and an agent named a that runs program P and is willing to access any of the r_i's. To control the access requests by the agent, one would typically refer to [DoD85] and set up a resource manager. In the Ambient Calculus the system under consideration can be represented as follows:

$$a[\![P]\!] \mid m[\![r_1[\![\cdots]\!] \mid \cdots \mid r_n[\![\cdots]\!] \mid R]\!]$$

Here, m is the resource manager running process R. To access, say r_i, the agent needs to know the name m, to be able to move inside the resource manager. Assuming the agent knows that name, the result of the move is the new system:

$$m[\![a[\![P]\!] \mid r_1[\![\cdots]\!] \mid \cdots \mid r_n[\![\cdots]\!] \mid R]\!]$$

Looking at this configuration, it is clear that the process R does not have an active role in the system: given the primitive constructs of the Ambient Calculus, there is indeed nothing R can do to enable or control the access, as the interaction between $a[\![P]\!]$ and each of the r_i's may only result from autonomous actions by either the agent or the resource[2]. The role of the ambient m is therefore reduced to the role of its name: it is simply the first password required for the access. Rather, it is each of the r_i's that needs to include its own manager.

We can thus formulate the problem in simpler terms, and look directly at the case of the agent a and the resource r shown below:

Initial configuration: $\quad a[\![P]\!] \mid r[\![R \mid \langle M \rangle]\!]$

R is the manager for r, and M is the contents: for the purpose of the example, we assume that the content is a value the agent is willing to read.

2.3 Overview of possible solutions

Having defined the problem, we now look at different ways to attack it in the Ambient Calculus, and discuss their implications for MAC security.

2.3.1. Agent dissolution. A first solution is based on the following protocol proposed by [CG98]. In order for a to access r, a first enters r:

Enter: $\quad r[\![R \mid \langle M \rangle \mid a[\![P]\!]]\!]$

[2] *Safe Ambients* [LS00] would not help here, as R would still be unable to mediate the access to r_i.

Now, the idea of the protocol is that the manager R should be the process $!\mathsf{open}\ p$, which unleashes authorized clients that entered the resource within a transport ambient named p. In other words, the protocol requires the client to know the name of the resource, as well the name of the "port" p used for access. The agent would first rename itself to p to comply with the rules of the protocol, and then enter: if the access to r is in read mode, the agent will contain a reading process. After renaming, the new configuration would then be:

Renaming: $r[\![\, !\mathsf{open}\, p \mid \langle M\rangle \mid p[\![\, (x)P \,]\!] \,]\!]$

Finally, the resource manager enables the read access, by opening p:

Read Access: $r[\![\, !\mathsf{open}\, p \mid \langle M\rangle \mid p[\![\, (x)P \,]\!] \,]\!] \rightarrow r[\![\, !\mathsf{open}\, p \mid \langle M\rangle \mid (x)P \,]\!]$

The protocol is elegant and robust, as the agent needs to know two passwords (r and p). There are, however, a number of unsatisfactory aspects to it.

A first reason for being unsatisfied with the protocol is that it is hardly realistic to assume that agents willing to read a value should be prepared to be dissolved. A second problem is that opening $p[\![\, P \,]\!]$ may be upsetting to the resource manager, or else to the resource itself, because there is no telling what P might do once unleashed. For what we know, the contents of p could very well be the process $N.P$, with N a path of in or out capabilities. Unleashing this process inside r could result into r being carried away to possibly hostile locations, or otherwise being made unavailable to other clients requesting access to it.

Further problems arise when we try to classify the protocol according to the principles of MAC security. As we noted, the action in the protocol that eventually enables the access to the resource is taken by the resource manager, which opens the incoming agent. In other words, it is the last step of the protocol that effectively determines the access, and since the process enclosed in p is an input process, it is classified as a read access (had p contained an output, it would have been a write access). In multilevel security, it would then be possible to further classify the access according to the security levels associated with r and p, and use that definition to enforce either the military or the commercial security policies.

However, while this form of classification is sensible for the protocol, it becomes rather artificial when applied to the primitives of the calculus. Indeed, saying that $\mathsf{open}\ p \mid p[\![\, P \,]\!]$ is a read (or write) access from P is rather counter-intuitive, as $p[\![\, P \,]\!]$ undergoes the action rather than actively participating into it. The problem is that the protocol is tightly dependent on the effects of open, but when exercised to enable a read/write request, open exchanges the roles of the two participants in the request, as it is the subject, rather than the object, that is accessed, in fact, opened.

2.3.2. Resource dissolution. The problem could be circumvented by a change of perspective. One could devise a different protocol where the active role of the subject is rendered by a combination of open and input/output. Thus, for instance, the process $\mathsf{open}\ r.(x)P$ could be interpreted, in the protocol, as a read request on r. This might work reasonably for read requests, even though the interpretation is not too convincing given that the access has also the side-effect of dissolving the resource. Even less convincing would be the interpretation of $\mathsf{open}\ r.\langle M\rangle$ as a write access: after dissolving r the output $\langle M\rangle$ really has nothing to do with a write on r.

2.3.3. Agents and messengers To avoid indiscriminate dissolution upon access, [CG98] suggests a different approach, based on a protocol similar to the first one we discussed, but in which agents rely on "special" ambients acting as messengers. The idea is to envisage two classes of messengers:

output messenger: $o[\![\, M.\langle N\rangle \,]\!]$. M is a path to the location where deliver message N;

input messengers: $i[\![\, M.(x)o[\![\, M^{-1}.\langle x\rangle \,]\!] \,]\!]$. M is the path to the location where a value can be read. Once read, the messenger goes back to its original location where it delivers the value.

Thus, a read access would be encoded by a protocol based on the following initial configuration:

$$a[\![\,\mathsf{open}\ o.(x)P \mid i[\![\,\mathsf{out}\ a.\mathsf{in}\ r.(x)\,o[\![\,\mathsf{out}\ r.\mathsf{in}\ a.\langle x\rangle\,]\!]\,]\!]\,]\!] \mid r[\![\,!\,\mathsf{open}\ i \mid \langle N\rangle\,]\!]$$

The protocol still requires cooperation from the resource manager, which is expected to open the input messenger. Also, looking at the primitive reductions, it would still be counter-intuitive to say that $\mathsf{open}\ i \mid i[\![\,P\,]\!]$ is a read access. However, if i could be identified as an input-messenger within r, then the access classification would be more realistic.

The problem is that there is no way to syntactically tell messengers from ambients playing the role of "pure" agents, nor is there any way to syntactically detect "illegal" attempts to dissolve "pure" agents. Defining a notion of access, and attempting a syntactic classification would therefore still be problematic, if at all possible.

Types could be appealed to for more satisfactory solution. One could devise a type system to complement the syntax by enforcing a typed partition of ambients into agents (i.e. ambients that cannot be dissolved) and messengers (as above). Based on the typed ambient classification and on an assignment of security levels, it would then be possible to classify access requests according to MAC policies. There would be only one remaining problem. Consider the protocol structure and evolution. From the initial configuration: $a[\![\,P' \mid i[\![\,M.(x)\,o[\![\,M^{-1}.\langle x\rangle\,]\!]\,]\!]\,]\!] \mid r[\![\,!\,\mathsf{open}\ i \mid \langle N\rangle\,]\!]$ a sequence of reductions routes the input messenger to its, where it it is opened and consumes N. At this stage, the structure of the system is: $a[\![\,P'\,]\!] \mid r[\![\,!\,\mathsf{open}\ i \mid o[\![\,M^{-1}.\langle N\rangle\,]\!]\,]\!]$. This is the encoding of a write access by r to a. In other words, a read access by a includes a write access by r: if the former is, say, a read-up, then the latter is a write-down. In other words, the protocol has somehow the effect of merging read-up's and write-down's, and dually, write-up's and read-down's. Therefore, military security could still be accounted for with this approach, while commercial security could not.

2.4 Summary and Assessment

The survey of solutions we have given may still be incomplete, but we do not see any significantly different approach to attack the problem. As to the approaches we have presented, none of them is fully adequate to reason about security. Some of them appear artificial, since essential intuition is lost in the encoding of the protocol (§ 2.3.1,§ 2.3.2), while in others, intuition is partially recovered but only at the expenses of failing to provide full account for both military and commercial security (§ 2.3.3).

Consequently, while possibly incomplete, the analysis does provide a basis for drawing a conclusion. Certainly, the Ambient Calculus *enables* resource access control, in that it provides constructs for encoding access protocols. On the other hand, the calculus *does not, by itself, support* these mechanisms and policies, as it does not provide built-in facilities to make it convenient or natural to reason about them. As we showed, the reasoning is possible at the level of access *protocols*, while when we look at the access *primitives*, there appears to be no general principle to which one can steadily appeal.

The conclusion we may draw, then, is that *support* for resource access control with Mobile Ambients requires different, finer-grained, constructs for ambient interaction and communication. The new constructs should be designed carefully, so as to complement the existing restrictions on ambient mobility based on authorization, without breaking them. In other words, access to remote resources should still require mobility, hence authorization: local access, instead, could be made primitive.

To see how that can be accomplished, consider once more the protocol of § 2.3.3, based on messengers. We can re-state it equivalently as follows:

$$a[\![\,\mathsf{in}\ r.\mathsf{open}\ o.(x)\mathsf{out}\ r.P \mid i[\![\,\mathsf{out}\ a.(x)\,o[\![\,\mathsf{in}\ a.\langle x\rangle\,]\!]\,]\!]\,]\!] \mid r[\![\,!\,\mathsf{open}\ i \mid \langle M\rangle\,]\!]$$

In other words, it is now the agent that is responsible for the moves needed to reach the resource, while the messenger just makes the in and out moves needed for the, now local, access. After the move of a into r, and of i out of a, the structure of the system (disregarding a) is the following: $r[\![\,\mathsf{open}\ i \mid \langle M\rangle \mid i[\![\,(x)P\,]\!]\,]\!]$. This is where the read access takes place. Now, instead of coding it, via open, we can make it primitive and do without open. If we denote with $(x)^{\uparrow}$ input from the enclosing ambient, the read access is simply: $r[\![\,\langle M\rangle \mid i[\![\,(x)^{\uparrow}P\,]\!]\,]\!]$. But then, the whole protocol can be simplified: $a[\![\,\mathsf{in}\ r.(x)^{\uparrow}.P\,]\!] \mid r[\![\,\langle M\rangle\,]\!]$.

A choice of communication primitives based on this observation led us to the design of *Boxed Ambients*, a calculus we formally define in [BCC01] and outline in the next section. The new primitives provide the calculus with what we believe to be more effective constructs for resource protection and access control, while at the same time retaining the expressive power and most of the computational flavor of Mobile Ambients, as well as the elegance of their formal presentation.

3 Boxed Ambients

Boxed Ambients are a variant of Cardelli and Gordon's Mobile Ambients. From the latter, they inherit the primitives in and out for mobility, with the exact same semantics. Instead, Boxed Ambients rely on a completely different model of communication, which results from dropping the open capability.

As in the Ambient Calculus, processes in the new calculus communicate via anonymous channels, inside ambients. In addition, to compensate for the absence of open, Boxed Ambients are equipped with primitives for communication across ambient boundaries, between parent and children. Syntactically, this is obtained by means of tags specifying the *location* where the communication has to take place. So for example, in $(x)^{n}P$ the input prefix $(x)^{n}$ is an input from child ambient n, while $\langle M\rangle^{\uparrow}$ is an output to the parent ambient.

The choice of these primitives is inspired to Castagna and Vitek's *Seal Calculus* [VC99], from which Boxed Ambients also inherit the two principles of *mediation* and *locality*. Mediation implies that remote communication, e.g. between sibling ambients, is not directly possible: it either requires mobility, or intervention by the ambients' parent. Locality means that communication resources are *local* to ambients, and message exchanges result from explicit read and write requests on those resources.

As it turns out, the resulting communication model has rather interesting payoffs when it comes to resource protection policies and security. Before entering further details, we briefly review the syntax and the semantics of the calculus.

Syntax and Semantics. The untyped syntax of the polyadic synchronous calculus is as follows.

$$\begin{array}{llcl} \textit{Expressions} & M & ::= & a, b, \ldots \mid x, y, \ldots \mid \mathsf{in}\ M \mid \mathsf{out}\ M \mid M.M \mid (M_1, \ldots, M_k) \\ \textit{Patterns} & \boldsymbol{x} & ::= & x \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \\ \textit{Locations} & \eta & ::= & M \mid \uparrow \mid \star \\ \textit{Processes} & P & ::= & \mathbf{0} \mid M.P \mid (\boldsymbol{\nu} x)P \mid P \mid P \mid M[\![\,P\,]\!] \mid !P \mid (\boldsymbol{x})^{\eta}P \mid \langle M\rangle^{\eta}P \end{array}$$

We use a number of notation conventions. We reserve $a - q$ for ambient names, and x, y, z for variables. As usual we omit trailing dead processes, writing M for $M.\mathbf{0}$. The superscript $\star$ denoting local communication, is also omitted.

The operational semantics is defined by reduction, with the help of an auxiliary relation of structural congruence. All these are very standard (in fact, exactly as in Ambient Calculus). We only give the top-level reduction rules, and refer the reader to [BCC01] for details.

Mobility. Reduction for the in and out capabilities is exactly as for Mobile Ambients:

$$\begin{array}{ll} (\textit{enter}) & a[\![\,\mathsf{in}\ b.P \mid Q\,]\!] \mid b[\![\,R\,]\!] \rightarrow b[\![\,a[\![\,P \mid Q\,]\!] \mid R\,]\!] \\ (\textit{exit}) & a[\![\,b[\![\,\mathsf{out}\ a.P \mid Q\,]\!] \mid R\,]\!] \rightarrow b[\![\,P \mid Q\,]\!] \mid a[\![\,R\,]\!] \end{array}$$

Communication. The primitives for local and parent-child communication are governed by the following rules. Note that in all cases input-output is synchronous (see § 4 for a brief digression on asynchronous communication).

(*local*)	$(x)P \mid \langle M \rangle Q \rightarrow P\{x := M\} \mid Q$
(*input n*)	$(x)^n P \mid n[\![\langle M \rangle Q \mid R]\!] \rightarrow P\{x := M\} \mid n[\![Q \mid R]\!]$
(*input* $\uparrow$)	$\langle M \rangle P \mid n[\![(x)^{\uparrow} Q \mid R]\!] \rightarrow P \mid n[\![Q\{x := M\} \mid R]\!]$
(*output n*)	$\langle M \rangle^n P \mid n[\![(x) Q \mid R]\!] \rightarrow P \mid n[\![Q\{x := M\} \mid R]\!]$
(*output* $\uparrow$)	$(x)P \mid n[\![\langle M \rangle^{\uparrow} Q \mid R]\!] \rightarrow P\{x := M\} \mid n[\![Q \mid R]\!]$

3.1 Resources and Access Control

Four different reductions for non-local exchange may be thought of as redundant, especially because there are only two reducts. Instead, different directions for input/output is a key design choice that has a number of interesting consequences.

- First, the primitives for communication have immediate and very natural interpretations as access requests. To exemplify, the input prefix $(x)^n$ can be seen as a request to read from the channel located into child ambient n. In fact, given the anonymous nature of channels, $(x)^n$ can equivalently be seen as an access to the ambient n. Dually, $\langle M \rangle^{\uparrow}$ can be interpreted as write request to the parent ambient (equivalently, its local channel)[3].
- Secondly, full and flexible support is now available for resource protection. An agent entering a resource needs not be opened there to enable the access: the resource manager can mediate and keep full control over the read and write requests made by the agent. If we take the resource access problem of § 2.2 we now have a fairly natural and elegant solution, and we also find back a role for the resource manager m. Consider again the configuration $m[\![\, a[\![P]\!] \mid r_1[\![\cdots]\!] \mid \cdots \mid r_n[\![\cdots]\!] \mid R \,]\!]$ where now all ambients are boxed, and a has entered the resource manager. We need not to include a manager in each resource, as R may act as a mediator. For instance, R could be defined as the parallel composition $R_1 \mid \cdots \mid R_n$ where each R_i is the process $!\,(x)\langle x \rangle^{r_i}$ waiting for upward output from a and forwarding it to the ith resource. Some of the R_i's could be less generous with the agent, and ignore upward input from a to request read access on a instead: $!\,(x)^a\langle x \rangle^{r_i}$. Should any of the r_i's be made non-accessible, one would simply define $R_i = \mathbf{0}$.
- The communication model fits nicely the security model of Mobile Ambients which is based on authorization and predicates in/out access to ambients on possession of appropriate passwords or cryptokeys.
- Finally, multilevel security for boxed ambients may be modeled by embedding security levels in types, and using typing rules to enforce and verify *Mandatory* (system-wide) *Access Control* (MAC) policies. We give a detailed account of how this can be done in § 3.2 below.

The calculus has other interesting aspects to it. For a thorough discussion on these aspects, and a detailed comparison between the communication primitives of Boxed and Mobile Ambients the reader is referred to [BCC01]. Here, instead, we focus our attention to security issues, and move on to multilevel security.

MAC Security. In MAC security, the behavior of system is described by a two-dimensional *Access Control Matrix* M indexed over a set S of *subjects* and a set O of *objects*, and whose

[3] The possibility to associate owners to channels is the reason why we do not consider *shared channels* in the style of [CGN01], that is, we do not have reductions such as, say,

$$(x)^n P \mid n[\![\langle M \rangle^{\uparrow} Q \mid R]\!] \rightarrow P\{x := M\} \mid n[\![Q \mid R]\!].$$

values are *access modes* $\mathscr{A},\mathscr{B} \in \{\mathsf{w},\mathsf{r},\mathsf{rw},\mathsf{shh}\}$. $\mathsf{M[s,o]}$=w (respectively r, rw, shh) indicates that subject s has write (respectively, read, read&write, no) access to object o.

For multilevel security, one presupposes a lattice $(\Sigma,\preceq)$ of *security levels* (ranged over by ρ,σ,τ), and a function *level* : $\mathsf{S} \cup \mathsf{O} \to \Sigma$. A *security policy* is a ternary boolean predicate $\mathscr{P}$ on subject levels, object levels, and access modes. An access control matrix M satisfies a security policy $\mathscr{P}$ if for every $\mathsf{s}\in\mathsf{S}$, $\mathsf{o}\in\mathsf{O}$, $\mathscr{P}(\mathit{level}(\mathsf{s}),\mathit{level}(\mathsf{o}),\mathsf{M[s,o]})$ holds true. Military (no read-up, no write-down) and commercial (no read-up, no write-up) security can then be formally defined as follows:

$$\begin{array}{llll}
\mathscr{P}_{Mil}(\rho,\sigma,\mathsf{r}) & \triangleq \sigma \preceq \rho & \qquad \mathscr{P}_{Com}(\rho,\sigma,\mathsf{r}) & \triangleq \sigma \preceq \rho \\
\mathscr{P}_{Mil}(\rho,\sigma,\mathsf{w}) & \triangleq \rho \preceq \sigma & \qquad \mathscr{P}_{Com}(\rho,\sigma,\mathsf{w}) & \triangleq \sigma \preceq \rho \\
\mathscr{P}_{Mil}(\rho,\sigma,\mathsf{rw}) & \triangleq \sigma = \rho & \qquad \mathscr{P}_{Com}(\rho,\sigma,\mathsf{rw}) & \triangleq \sigma \preceq \rho \\
\mathscr{P}_{Mil}(\rho,\sigma,\mathsf{shh}) & \triangleq \mathit{true} & \qquad \mathscr{P}_{Com}(\rho,\sigma,\mathsf{shh}) & \triangleq \mathit{true}
\end{array}$$

EXCURSUS. *In process algebras, it is interesting to take a powerset of security labels* $(2^L,\subseteq)$ *as lattice of security levels. Based on that, it is possible to use standard* π*-calculus restrictions to dynamically define security levels:* $(\boldsymbol{\nu}\ell\colon L)(\boldsymbol{\nu}x\colon\{\ell\})P$*. New formalizations of* Discretionary Access Control *policies (DAC) are then possible if, in addition, one also allows security labels to be communicated over channels. We discuss this possibility with a brief* aperçu *in Section 4.*

3.2 A Type System for MAC Multilevel Security

The type system results from a rather simple refinement of the type system for Boxed Ambients defined in [BCC01]. As in that case, ambient and process types are defined as two-place constructors describing the types of exchanges that may occur locally and with the enclosing context. Interestingly, this simple type structure is all that is needed to give a full account of ambient interaction. This is a consequence of (i) there being no way for ambients to communicate directly across more than one boundary, and (ii) communication being the only means for ambient to interact.

Multilevel security is accounted for in the type system by instrumenting the structure of types to include additional information about the security level associated with each ambient (viewed as subject or object) and the access mode of the ambient's exchange types. The resulting syntax of types, as well as its intended meaning, are defined as follows, where the metavariable $\mathscr{A}$ ranges over access modes:

$$\begin{array}{lrcl}
\textit{Expression Types} & W & ::= & \sigma\mathsf{Amb}[E,F^{\mathscr{A}}] \mid \sigma\mathsf{Cap}[E^{\mathscr{A}}] \mid W_1 \times \cdots \times W_n \\
\textit{Exchange Types} & E,F & ::= & \mathsf{shh} \mid W \\
\textit{Process Types} & T & ::= & \sigma\mathsf{Pro}[E,F^{\mathscr{A}}]
\end{array}$$

Ambient types $\sigma\mathsf{Amb}[E,F^{\mathscr{A}}]$: the type of ambients with clearance σ, enclosing processes whose local and upward exchanges are of type E and F; the upward exchanges have mode $\mathscr{A}$.

Capability types $\sigma\mathsf{Cap}[F^{\mathscr{A}}]$: the type of capabilities exercised within an ambient of clearance σ, whose upward exchanges have type F and mode $\mathscr{A}$.

Process types $\sigma\mathsf{Pro}[E,F^{\mathscr{A}}]$: the type of processes running at clearance σ, whose local and upward exchanges are, respectively, of type E and F. The tag $\mathscr{A}$ defines the mode in which the process accesses the channel located in its parent ambient.

In all cases, the type shh indicates no exchange, that is, absence of input *and* output. The syntax allows the formation of the types $\mathsf{Amb}[E,\mathsf{shh}^{\mathscr{A}}]$, $\mathsf{Cap}[\mathsf{shh}^{\mathscr{A}}]$, and $\mathsf{Pro}[E,\mathsf{shh}^{\mathscr{A}}]$. These types are convenient in stating definitions and typing rules: to make sense of them, we stipulate that $\mathsf{shh}^{\mathscr{A}} = \mathsf{shh}$ for any access $\mathscr{A}$.

To enhance the flexibility of the type system, we introduce the following subtype relation over exchange types.

Definition 1 (*Exchange Subtyping). Let $\leqslant$ be the smallest reflexive and transitive relation over exchange types satisfying the following axioms for every exchange type E and access mode $\mathscr{A}$:*
$\mathsf{shh} \leqslant E, \quad \mathsf{shh} \leqslant E^{\mathscr{A}}, \quad E^{\mathsf{r}} \leqslant E^{\mathsf{rw}}, \quad E^{\mathsf{w}} \leqslant E^{\mathsf{rw}}$ □

Exchange subtyping is not used in conjunction with subsumption. Subtyping may be lifted to capability and process types to allow sound uses of subsumption. This enhanced form of subtyping is studied in [BCC01], but is essentially orthogonal to the subject of our present discussion. We therefore disregard it, and move on to illustrate the typing rules.

The typing rules are presented in Figure 1, and discussed next. The rules (IN) and (OUT) define the constraints for ambient mobility. They explain why capability types are built around a single component, and motivate the subtyping relation over exchange types. The intuition of the (IN) is as follows: if $\mathsf{Cap}[F]$ is the type of the capability, say $\mathsf{in}\ n$, then $\mathsf{in}\ n$ is exercised within an ambient, say m, with upward exchanges F. Now, for the move of m into n to be safe, one must ensure that the local exchanges of n also have type F. In fact, one may be more liberal, and only require type compatibility between the upward exchanges of m and the local exchanges of n: this explains the premise $E \leqslant G$. The predicate $\mathscr{P}$ provides a guarantee that after moving, the ambient will still satisfy the security policy. Dual reasoning applies to the (OUT) rule: upward exchanges by the exiting ambient must have the same (in fact, $\leqslant$-compatible) type as the upward exchanges of the ambient being exited. The security policy is enforced, in this case, directly by the subtyping relation over exchange types. It is worth noting that upward silent ambients (that is, ambients whose upward exchanges have type shh) can freely move across ambient boundaries. This is a consequence of our interpretation of capabilities, and of how $\leqslant$ is defined: capabilities exercised within upward silent ambients have type $\sigma\mathsf{Cap}[\mathsf{shh}]$ and $\mathsf{shh} \leqslant E$ for every E.

The rule (AMB), for typing ambients, defines the constraints that must be satisfied by P to legally be enclosed in a: specifically, the type of the upward exchanges performed by P, must comply with the security policy defined by the predicate $\mathscr{P}$ and must be a subtype of the local exchanges of the current ambient (that is either shh or G). As an example, if P tries to read from the channel located in the ambient that encloses a, then, to avoid a *read up* operation, the clearance of P (i.e. that of the ambient a) must be higher than that of the accessed channel (i.e. that of the ambient enclosing a).

The other interesting rules are those for communication. Local communication, i.e. local access within an ambient, needs no security constraint. The rules (INPUT M) and (OUTPUT M) govern input/output to subambients. Besides connecting the types of the input-output processes and their continuations, the rules also enforce the constraints that processes at clearance σ read only from (resp. write only to) ambients of clearance ρ compatible with σ according to the given security policy.

We conclude with the (INPUT $\uparrow$) and (OUTPUT $\uparrow$) for upward input/output which, perhaps surprisingly, do not impose any security constraint: that is because security on upward communication is already regulated by the ambient rule, and by the rules governing mobility.

The type system satisfies standard properties, notably, Subject Reduction:

Theorem 1. *If $\Gamma \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]$ and $P \rightarrow Q$, then $\Gamma \vdash Q : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]$.*

However, the main purpose of types is to statically detect access violations. It is a simple technical matter to show the soundness of our type system. Let *level* be the function that associates the types $\sigma\mathsf{Amb}[E, F^{\mathscr{A}}]$ and $\sigma\mathsf{Cap}[E^{\mathscr{A}}]$ to σ. We decorate reduction with a function ℓ that associate names to security levels. The definition is straightforward in all cases, except for the case of restrictions:

Typing of Expressions

(PROJECT)
$$\frac{\Gamma(n) = W}{\Gamma \vdash n : W}$$

(TUPLE)
$$\frac{\Gamma \vdash M_i : W_i \quad \forall i \in 1..k}{\Gamma \vdash (M_1, \ldots, M_k) : W_1 \times \cdots \times W_k}$$

(PATH)
$$\frac{\Gamma \vdash M_i : \sigma\mathsf{Cap}[E^{\mathscr{A}}] \quad i = 1, 2}{\Gamma \vdash M_1.M_2 : \sigma\mathsf{Cap}[E^{\mathscr{A}}]}$$

(IN)
$$\frac{\Gamma \vdash M : \rho\mathsf{Amb}[G, H^{\mathscr{B}}] \quad \mathscr{P}(\sigma, \rho, \mathscr{A}) \quad E \leqslant G}{\Gamma \vdash \mathsf{in}\ M : \sigma\mathsf{Cap}[E^{\mathscr{A}}]}$$

(OUT)
$$\frac{\Gamma \vdash M : \rho\mathsf{Amb}[G, H^{\mathscr{B}}] \quad E^{\mathscr{A}} \leqslant H^{\mathscr{B}}}{\Gamma \vdash \mathsf{out}\ M : \sigma\mathsf{Cap}[E^{\mathscr{A}}]}$$

Typing of Processes

(PREFIX)
$$\frac{\Gamma \vdash M : \sigma\mathsf{Cap}[F^{\mathscr{A}}] \quad \Gamma \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}{\Gamma \vdash M.P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(PARALLEL)
$$\frac{\Gamma \vdash P_i : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}] \quad i = 1, 2}{\Gamma \vdash P_1 \mid P_2 : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(AMB)
$$\frac{\Gamma \vdash a : \sigma\mathsf{Amb}[E, F^{\mathscr{A}}] \quad \Gamma \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}] \quad \mathscr{P}(\sigma, \rho, \mathscr{A}) \quad F \leqslant G}{\Gamma \vdash a[\![\, P \,]\!] : \rho\mathsf{Pro}[G, H^{\mathscr{B}}]}$$

(INPUT $\star$)
$$\frac{\Gamma, x : W \vdash P : \sigma\mathsf{Pro}[W, F^{\mathscr{A}}]}{\Gamma \vdash (x : W)P : \sigma\mathsf{Pro}[W, F^{\mathscr{A}}]}$$

(OUTPUT $\star$)
$$\frac{\Gamma \vdash M : W \quad \Gamma \vdash P : \sigma\mathsf{Pro}[W, F^{\mathscr{A}}]}{\Gamma \vdash \langle M \rangle P : \sigma\mathsf{Pro}[W, F^{\mathscr{A}}]}$$

(INPUT $\uparrow$) $\quad \mathscr{A} \in \{\mathsf{r}, \mathsf{rw}\}$
$$\frac{\Gamma, x : W \vdash P : \sigma\mathsf{Pro}[E, W^{\mathscr{A}}]}{\Gamma \vdash (x : W)^{\uparrow} P : \sigma\mathsf{Pro}[E, W^{\mathscr{A}}]}$$

(OUTPUT $\uparrow$) $\quad \mathscr{A} \in \{\mathsf{w}, \mathsf{rw}\}$
$$\frac{\Gamma \vdash M : W \quad \Gamma \vdash P : \sigma\mathsf{Pro}[E, W^{\mathscr{A}}]}{\Gamma \vdash \langle M \rangle^{\uparrow} P : \sigma\mathsf{Pro}[E, W^{\mathscr{A}}]}$$

(INPUT M)
$$\frac{\Gamma, x : W \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}] \quad \Gamma \vdash M : \rho\mathsf{Amb}[W, U^{\mathscr{B}}] \quad \mathscr{P}(\sigma, \rho, \mathsf{r})}{\Gamma \vdash (x : W)^{M} P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(OUTPUT M)
$$\frac{\Gamma \vdash N : W \quad \Gamma \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}] \quad \Gamma \vdash M : \rho\mathsf{Amb}[W, U^{\mathscr{B}}] \quad \mathscr{P}(\sigma, \rho, \mathsf{w})}{\Gamma \vdash \langle N \rangle^{M} P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(DEAD)
$$\frac{}{\Gamma \vdash \mathbf{0} : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(REPLICATION)
$$\frac{\Gamma \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}{\Gamma \vdash !P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

(NEW)
$$\frac{\Gamma, x : W \vdash P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}{\Gamma \vdash (\boldsymbol{\nu} x : W)P : \sigma\mathsf{Pro}[E, F^{\mathscr{A}}]}$$

Fig. 1. Typing Rules

$$\frac{P \;\rightarrow_{\ell,(x\,:\,\mathit{level}(W))}\; Q}{(\boldsymbol{\nu} x : W)P \;\rightarrow_{\ell}\; (\boldsymbol{\nu} x : W)Q}$$

Also, we instrument this form of labeled reduction with *error rules*.

$$\begin{array}{lll}
(\textit{e-input } n) & m[\![(x{:}W)^n P \mid n[\![\langle M\rangle Q \mid R]\!] \mid S]\!] \rightarrow_\ell \mathtt{err} & \textit{if} \;\; \neg\mathscr{P}(\ell(m), \ell(n), \mathsf{r}) \\
(\textit{e-input } \uparrow) & m[\![\langle M\rangle P \mid n[\![(x)^\uparrow Q \mid R]\!] \mid S]\!] \rightarrow_\ell \mathtt{err} & \textit{if} \;\; \neg\mathscr{P}(\ell(n), \ell(m), \mathsf{r}) \\
(\textit{e-output } n) & m[\![\langle M\rangle^n P \mid n[\![(x)Q \mid R]\!] \mid S]\!] \rightarrow_\ell \mathtt{err} & \textit{if} \;\; \neg\mathscr{P}(\ell(m), \ell(n), \mathsf{w}) \\
(\textit{e-output } \uparrow) & m[\![(x)P \mid n[\![\langle M\rangle^\uparrow Q \mid R]\!] \mid S]\!] \rightarrow_\ell \mathtt{err} & \textit{if} \;\; \neg\mathscr{P}(\ell(n), \ell(m), \mathsf{w})
\end{array}$$

In addition, we have structural rules that propagate errors from a process to its enclosing terms. Finally, given a type environment Γ, we say that ℓ is Γ-compatible if for all $x \in \mathit{dom}(\Gamma)$, one has $\ell(x) = \mathit{level}(\Gamma(x))$. If we assume that `err` is a distinguished process, with no type, it is very easy to verify that no system containing an occurrence of `err` can be typed in our type system. Absence of run time errors may now be stated as follows:

Theorem 2 (Soundness). *For every Γ, P and Γ-compatible ℓ, if $\Gamma \vdash P : T$, then $P \not\rightarrow_\ell$* `err`.

4 Examples

In this section we consider several examples from the literature on security and related issues, and show how to handle them with Boxed Ambients.

Wrappers. As a solution for resource protection and access control in wide-area networks, Sewell and Vitek [SV00] propose to use *wrappers* to isolate potentially malicious programs. Their framework is based on an extension of the π-calculus, known as the *boxed* π-calculus: *wrappers* enable a programming style in which incoming code can be secured into a *box*, and its interactions with the enclosing environment filtered by the *wrapper* that only forwards legitimate messages between the boxed program and its enclosing environment via secured channels.

The paradigmatic example of that work can be rephrased in our syntax as follows:

$$(\nu a, b)\,\big(a[P] \mid !(x)^a \langle x\rangle^b \mid b[Q]\big)$$

P and Q are arbitrary processes encapsulated in ambients ("named boxes" in [SV00] terminology) with private names a and b, placed in parallel with a forwarder process from ambient a to ambient b. The configuration above is interesting when P and Q are distrusted processes since ambient boundaries forbid them to interact directly, while the restrictions ensure that the only possible interaction with the environment is with the forwarder process $!(x)^a\langle x\rangle^b$. This is the way for Boxed-π to enforce a security policy that prevents (*i*) Q from leaking secrets to P and (*ii*) P and Q from corrupting the environment. This holds true also in Boxed Ambients. Besides that, in Boxed Ambients we have the choice of other alternatives. For example, to enforce (*i*) we can use military security and ensure a more general property: if we assign to a a security level strictly greater than the level of b, then our type system statically ensures that there cannot be any unwanted access from Q to P. To enforce also (or only) the property (*ii*) we can once more rely on military (but also commercial) security, and assign to the environment a security level incomparable with the levels of a and b. Then the two processes cannot access and corrupt the resources of the environment.

Asynchronous Communication In wide-area networks it is hardly reasonable to rely only on synchronous communication (see [Car00] for discussion, and [BV02] for experience with implementations). In [BCC01], we show how to account for asynchronous output in Boxed Ambients and discuss the consequences of this choice. Besides other changes, asynchronous communication results from introducing the following new reduction rules

$$\begin{array}{ll}(\textit{asynch output } n) & \langle M\rangle^{n} P \mid n[\![\, Q \,]\!] \rightarrow P \mid n[\![\, \langle M\rangle \mid Q \,]\!] \\ (\textit{asynch output } \uparrow) & n[\![\, \langle M\rangle^{\uparrow} P \mid Q \,]\!] \rightarrow \langle M\rangle \mid n[\![\, P \mid Q \,]\!]\end{array}$$

to direct an output in the appropriate ambient. The enhanced flexibility obtained using asynchronous communications is paid by lesser security since now we loose the total mediation principle. Consider the following two examples:

$$a[\![\, (x : W)^{b} P \mid b[\![\, c[\![\, \langle M\rangle^{\uparrow} \mid Q \,]\!] \,]\!] \,]\!] \qquad b[\![\, a[\![\, (x : W)^{\uparrow} P \,]\!] \mid c[\![\, \langle M\rangle^{\uparrow} Q \,]\!] \,]\!]$$

they both implement a *covert channel* between ambients a and c, since with asynchronous reductions, they evolve into $a[\![\, (x{:}W)^{n} P \mid b[\![\, \langle M\rangle \mid c[\![\, Q \,]\!] \,]\!] \,]\!]$ and $b[\![\, a[\![\, (x{:}W)^{\uparrow} P \,]\!] \mid \langle M\rangle \mid c[\![\, Q \,]\!] \,]\!]$ respectively. In both cases by a further reduction step the ambient a gets hold of the message $\langle M\rangle$ without any mediation of b.

These kind of covert channels are two examples of security breaches that cannot be prevented by the primitives of the calculus and it is where the use of security policies comes to rescue. In both cases it just suffices to assign to ambient a a clearance strictly lower than that of b to make the read operation performed by a illegal in both commercial and military security (since it would be a read-up) and, as such, statically detected.

Firewalls We now look at the protocol for firewall crossing defined in [CG99] and refined in [LS00], and show how it can be defined with Boxed Ambients. The idea of the protocol is to let an *Agent* cross a *Firewall* by means of a shared key k.

$$\textit{Firewall} = (\boldsymbol{\nu} f) f[\![\, k[\![\, \mathsf{out}\ f.\langle \mathsf{in}\ f\rangle^{a} \,]\!] \mid \ldots \,]\!] \qquad \textit{Agent} = a[\![\, \mathsf{in}\ k.(x)\mathsf{out}\ k.x.Q \,]\!]$$

$$\begin{array}{lll}(\boldsymbol{\nu} k)(\textit{Firewall} \mid \textit{Agent}) & \rightarrow^{*} & (\boldsymbol{\nu} k, f) f[\![\, \ldots \,]\!] \mid k[\![\, \langle \mathsf{in}\ f\rangle^{a} \mid a[\![\, (x)\mathsf{out}\ k.x.Q \,]\!] \,]\!] \\ & \rightarrow^{*} & (\boldsymbol{\nu} k, f) f[\![\, \ldots \,]\!] \mid k[\![\, a[\![\, \mathsf{out}\ k.\mathsf{in}\ f.Q \,]\!] \,]\!] \\ & \rightarrow^{*} & (\boldsymbol{\nu} f) f[\![\, \ldots \mid a[\![\, Q \,]\!] \,]\!]\end{array}$$

The *Firewall*, with secret name f, sends out a pilot ambient k to guide the agent inside. The name k is a password that the agent a must know in order to enter (to acquire the path to) the firewall.

Besides authenticating entering agents, the firewall must in general provide other security guarantees. For example, the firewall administrator may want to ensure that processes inside the firewall can access the resources of an entered agent, but not the converse. This can be enforced with commercial security, by the following type assignments: $f : \phi\mathsf{Amb}[E, F^{\mathscr{A}}]$ and $k : \kappa\mathsf{Amb}[\mathsf{shh}, \mathsf{shh}]$, where E and $F^{\mathscr{A}}$ are appropriate types, and ϕ and κ are security levels such that $\kappa \prec \phi$. To illustrate the effects of this type assignments, consider a generic agent a (whose definition may differ from that of *Agent*) that wants to enter the firewall, and assume that $a : \alpha\mathsf{Amb}[G, H^{\mathscr{B}}]$. To cross the firewall, a must comply with the protocol and therefore accept write requests from k. With commercial security, this is possible only if $\alpha \preceq \kappa$ and this, by transitivity, implies that $\alpha \prec \phi$. Moreover, commercial security forbids low-level ambients (such as a) contained in high-level ambients (such as f) to perform upward communications. But then, $\alpha \prec \phi$ implies $\mathscr{B} = \mathsf{shh}$. In summary, the type assignment enforces for a a security level strictly smaller than the level of f, and the policy we choose ensures that the agents that enter the firewall f cannot directly access to local resources of f, as expected.

The protocol we just discussed depends on the assumption that the firewall knows the name of the entering agents. This is clearly unrealistic but, fortunately, easily remedied, as we show next. In order to provide guarantees of commercial security, the new protocol assumes that the agent know two passwords, k_1 and k_2, to cross the firewall.

$$\begin{array}{ll}\textit{Firewall}_2 & = (\boldsymbol{\nu} f) f[\![\, k_1[\![\, \mathsf{out}\ f.(y{:}A)^{k_2} \mathsf{in}\ f.\langle N\rangle^{y} \langle y\rangle \,]\!] \mid (y{:}A)^{k_1} P\{y\} \,]\!] \\ \textit{Agent}_2 & = a[\![\, \mathsf{in}\ k_1.\,(k_2[\![\, \mathsf{out}\ a.\langle a\rangle \,]\!] \mid (x)\mathsf{out}\ k_1.Q) \,]\!]\end{array}$$

$(\boldsymbol{\nu} k, k')(\mathit{Firewall}_2 \mid \mathit{Agent}_2)$

$$\rightarrow^* \quad (\boldsymbol{\nu} k_1, k_2, f) f[\![\ldots]\!] \mid k_1 [\![(y{:}A)^{k_2} \mathsf{in}\ f.\langle N\rangle^y \langle y\rangle \mid k_2 [\![\langle a\rangle]\!] \mid a[\![(x) \mathsf{out}\ k_1.Q]\!]]\!]$$

$$\rightarrow^* \quad (\boldsymbol{\nu} f) f[\![a[\![Q]\!] \mid P\{a\}]\!]$$

Again, the protocol starts with the agent a entering the pilot ambient k_1. The ambient k_1, in turn, reads from k_2 the name of the agent, carries the agent inside the firewall, and communicate the name a in order to let the firewall interact with the incoming agent. The message $\langle N\rangle^y$ is just used for synchronization, to ensure that the agent a exits the pilot ambient k_2 only after k_2 is back into the firewall. Note that the firewall may interact only with agents whose type is the one used for the variable y. It would be nice to add to subtyping (and tuple?) polymorphism other forms of polymorphism (for example on the lines of the excellent [AKPG01]) so that to extend the possible interactions with the incoming agents.

Trojan Horses In [BC01] a type system that can statically detect Trojan horses is defined. The motivating example is the system $a[\![\mathsf{in}\ c.P]\!] \mid b[\![\mathsf{in}\ a.\mathsf{out}\ a.\mathsf{in}\ d.Q]\!] \mid c[\![R \mid d[\![S]\!]]\!]$, where the ambient d contains confidential data that should be made available to ambients running within c but not to ambients that will enter c. The question is whether c should let a enter or not. Apparently a does not attempt to access d; nevertheless the move must be forbidden since b can use it as a Trojan Horse to enter c ad then access d.

$$a[\![\mathsf{in}\ c.P]\!] \mid b[\![\mathsf{in}\ a.\mathsf{out}\ a.\mathsf{in}\ d.Q]\!] \mid c[\![R \mid d[\![S]\!]]\!] \quad \rightarrow^* \quad c[\![R \mid a[\![P]\!] \mid d[\![S \mid b[\![Q]\!]]\!]]\!]$$

The attack is detected in [BC01] by means of a type system that traces the behavior of a, revealing the move of b into a and hence a chance for the attack. In [BC01] it is also shown how to perform this verification even when c runs in a possibly untyped context. Here we can obtain the same effect by setting the clearance of d to a level that is incomparable to any security level that is defined outside c. As we hinted in the excursus in Section 3.1 this can be obtained by using security labels with limited scope: $(\boldsymbol{\nu}\ell : L)(\boldsymbol{\nu} d : \{\ell\}\mathsf{Amb}[E, \mathsf{shh}])c[\![R \mid d[\![S]\!]]\!]$. No matter how and where the name d is communicated and whether c is in a well typed-context or not, if we impose commercial security only the processes that are already inside the ambient c can access information contained in d. Indeed to reproduce the initial configuration, c must communicate to b the name d. But, unlike what happens in Mobile Ambients, revealing the name of an ambient does not imply granting access to its resources.

5 Related work and conclusions

We have studied the problem of MAC security for Mobile Ambients (MA), and argued that the calculus is not fully adequate to express security concepts. As a solution, we have presented Boxed Ambients (BA), whose primitives provide elegant and natural mechanisms of resource access control. We conclude with discussion on related work on security for calculi of mobility.

The Dπ Calculus. In [HR00b] Hennessy and Riley discuss a type system for resource protection in the $D\pi$-calculus, a distributed variant of π-calculus where processes are located, and may migrate across locations. In Dπ, communication occurs via named channels that are associated with read/write capabilities: the type system controls that processes accessing a resource possess the appropriate capability.

In our approach, instead, in order to classify an access as legal or illegal, the type system checks that the security levels of subject and object satisfy the constraints imposed by the security policy for that access. A further difference is that in $D\pi$ the topology of locations is completely flat, while in BA ambients may be nested at will: the interplay between the dynamic nesting structure determined by moves, and the dynamic binding of the parent location $\uparrow$ for upward communication makes access control for BA more complex. In [**?**], the type system for Dπ is extended to cope with *partially typed networks*, in which some of the agents (and/or locations) are untyped, hence untrusted: type safety for such networks requires a form of dynamic type checking. Plans for future work on BA include extensions along similar lines.

The Security Pi calculus. In [HR00a], Hennessy and Riely discuss the *security π-calculus*, a variant of the π-calculus in which processes are syntactically defined as running at a given security level, and whose type system ensures that low-level processes never gain access to high-level resources. In BA, instead, we assume that clearances are specified by types, and the security level associated to an ambient type represents the clearance of resources contained in that ambient, as well as the clearance of the agent it implements. Besides resource protection, in [HR00a], the authors also investigate non-interference, trying to provide guarantees against implicit information flow from high levels to lower levels. To that end, they check that the clearance of values are compatible with clearance of channels along which they (the values) are exchanged. Furthermore, they show that a suitable notion of observational equivalence (*may test*) is soundly captured by the non-interference property checked by the type system.

We did not study these issues in detail (but see § 4 for examples of information flow) in this paper. In fact, in its current version, the type system only checks the clearance of subjects against the clearance of objects, disregarding the clearance of the values. We believe that it is possible to study BA in a similar way taking these issues into account. We leave this and the study of information flow and non-interference as subject of future work.

Typing of Mobility and Security for Mobile Ambients. Our type system is clearly related to other typing systems developed for Mobile Ambients. In [CG99] types guarantees absence of type confusion for communications. The type systems of [CGG99] and [Zim00] provide control over ambients moves and opening. Furthermore, the introduction of *group* names [CGG00] and the possibility of creating fresh group names, give flexible ways to statically prevent unwanted propagation of names. The powerful type discipline for Safe Ambients, presented in [LS00], adds finer control over ambient interactions and prevents all *grave interferences*.

All these approaches are orthogonal to the resource access control mechanisms we studied. We believe that similar typing disciplines as well as the use of group names, can be adapted to Boxed Ambients to obtain similar strong results. A paper more directly related to ours is [DCS00], where ambient types are associated with security levels in ways similar to ours. The difference is that in [DCS00], security checks are over opening and moves, while in our work we focus on read and write operations.

A final mention goes to [BC01], [NNHJ99] and [NN00], where control and data flow analysis, rather than typing disciplines, are used to check security properties of Ambients.

Other Languages and Calculi. In the literature on language-based security, several papers deal with access control techniques and information flow. Two examples are [DNFP99] and [LR99]. In [DNFP99], authors take a similar approach to that of Hennessy and Riley, based on a variant of Linda with multiple "tuple spaces". In [LR99], Leroy and Rouaix focus on integrity of typed applets via access control.

Acknowledgments. Thanks to Luca Cardelli for insightful comments and discussion, and to the anonymous referees. The second author would like to thank Ilaria Castagna for several corrections she made on an early draft of the paper.

References

[AKPG01] T. Amtoft, A.J. Kfoury, and S.M. Pericas-Geertsen. What are polymorphically-typed ambients? In *ESOP 2001*, volume 2028 of *Lecture Notes in Computer Science*, pages 206–220. Springer, 2001.

[BC01] M. Bugliesi and G. Castagna. Secure safe ambients. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages*, pages 222–235, London, 2001. ACM Press.

[BCC01] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. Technical report, L.I.E.N.S., 2001. Available at ftp.ens.fr/pub/dmi/users/castagna.

[BP76] D.E. Bell and L. La Padula. Secure computer system: Unified exposition and multics interpretation,. Technical Report MTR-2997, MITRE Corporation, Bedford, MA 01730, March 1976.

[BV02] C. Bryce and J. Vitek. The JavaSeal mobile agent kernel. *Autonomous Agents and Multi-Agent Systems*, 2002. To appear.

[Car00] L. Cardelli. Global computing. In *IST FET Global Computing Consultation Workshop*. 2000. Slides.

[CG98] L. Cardelli and A. Gordon. Mobile ambients. In *Proceedings of POPL'98*. ACM Press, 1998.

[CG99] L. Cardelli and A. Gordon. Types for mobile ambients. In *Proceedings of POPL'99*, pages 79–92. ACM Press, 1999.

[CGG99] L. Cardelli, G. Ghelli, and A. Gordon. Mobility types for mobile ambients. In *Proceedings of ICALP'99*, number 1644 in Lecture Notes in Computer Science, pages 230–239. Springer, 1999.

[CGG00] L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In *International Conference IFIP TCS*, number 1872 in Lecture Notes in Computer Science, pages 333–347. Springer, August 2000.

[CGN01] G. Castagna, G. Ghelli, and F. Zappa Nardelli. Typing mobility in the seal calculus. In *CONCUR 2001 (12th. International Conference on Concurrency Theory)*, Lecture Notes in Computer Science, Aahrus, Danemark, 2001. Springer. This same volume.

[DCS00] M. Dezani-Ciancaglini and I. Salvo. Security types for safe mobile ambients. In *Proceedings of ASIAN'00*, pages 215–236. Springer, 2000.

[DNFP99] R. De Nicola, G. Ferrari, and R. Pugliese. Types as specifications of access policies. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in LNCS. Springer, 1999.

[DoD85] US Department of Defense. Dod trusted computer system evaluation criteria, (the orange book). DOD 5200.28-STD, 1985.

[FGL$^+$96] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR'96)*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 1996.

[Gol99] D. Gollmann. *Computer Security*. John Wiley & Sons Ltd., 1999.

[HR00a] M. Hennessy and J. Riely. Information flow vs. resource access in the asynchronous π-calculus (extended abstract). In *Automata, Languages and Programming, 27th International Colloquium*, volume 1853 of *Lecture Notes in Computer Science*, pages 415–427. Springer, 2000.

[HR00b] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 2000. To appear.

[LR99] X. Leroy and F. Rouaix. Security properties of typed applets. In J. Vitek and C. Jensen, editors, *Secure Internet Programming – Security issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*, pages 147–182. Springer, 1999.

[LS00] F. Levi and D. Sangiorgi. Controlling interference in Ambients. In *POPL '00*, pages 352–364. ACM Press, 2000.

[NN00] H. R. Nielson and F. Nielson. Shape analysis for mobile ambients. In *POPL'00*, pages 135–148. ACM Press, 2000.

[NNHJ99] F. Nielson, H. Riis Nielson, R. R. Hansen, and J. G. Jensen. Validating firewalls in mobile ambients. In *Proc. CONCUR'99*, number 1664 in LNCS, pages 463–477. Springer, 1999.

[SV00] P. Sewell and J. Vitek. Secure composition of untrusted code: Wrappers and causality types. In *13th IEEE Computer Security Foundations Workshop*, 2000.

[VC99] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, number 1686 in Lecture Notes in Computer Science. Springer, 1999.

[Zim00] P. Zimmer. Subtyping and typing algorithms for mobile ambients. In *Proceedins of FoS-SaCS'99*, volume 1784 of *LNCS*, pages 375–390. Springer, 2000.