# Information Leakage Detection in Boundary Ambients ⋆

Chiara Braghin [1], Agostino Cortesi [2] and Riccardo Focardi [3]

*Dipartimento di Informatica, Università Ca' Foscari di Venezia, via Torino 155, 30172 Venezia – Mestre (Italy)*

**Abstract**

A variant of Mobile Ambient Calculus is introduced, called Boundary Ambient, to model multilevel security policies. Ambients that may guarantee to properly protect their content are explicitly identified as *boundaries*: a boundary can be seen as a resource access manager for confidential data. In this setting, we define a notion of non-interference which captures the absence of any (both direct and indirect) information leakage. Then, we guarantee non-interference by extending a control flow analysis that computes an over approximation of all ambients and capabilities that may be affected by the actual values of high level data.

## 1 Introduction

Mobile Ambient Calculus [12] has become a very interesting workbench to reason about mobility related issues, in which security plays a crucial role.

Our starting point is the "pure" version of Mobile Ambients, in which no communication channels are present and the only possible actions are represented by the moves performed by mobile processes, as it allows the study of a very general notion of information flow which should be applicable also to more "concrete" versions of the calculus (see, e.g., [10,25]).

In this setting, we are interested in focusing on *Multilevel Security* [2], a particular *Mandatory Access Control* security policy: every entity is bound to a security level (high and low, for simplicity), and information may only flow from the low to the high level. Typically, two access rules are imposed: *No*

⋆ Partially supported by MIUR Projects "Interpretazione Astratta, Type Systems e Analisi Control-Flow" and "Modelli formali per la sicurezza", and the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems".

[1] Email:braghin@dsi.unive.it

[2] Email:cortesi@dsi.unive.it

[3] Email:focardi@dsi.unive.it

*This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science*
*URL:* www.elsevier.nl/locate/entcs

*Read Up* (a low level entity cannot access information of a high level entity) and *No Write Down* (a high level entity cannot leak information to a low level entity). Sometimes, these two access controls are not enough as information may be indirectly leaked, through, e.g., some system side-effects: a typical example is represented by a resource shared among the security levels which may be alternatively overloaded by some Trojan horse (causing, e.g., longer response time at all security levels) in order to transmit information to a malicious low level entity.

In [4,6], we introduced the notion of *security boundary*, that allows us to identify ambients that may guarantee to properly protect their content. The intuition is the following: to guarantee absence of information leakage, every high-level data or process should be encapsulated into a boundary, and a boundary can be opened only when it is nested into another boundary. We may even allow that low level information/processes interfere with high level data inside a boundary, but we want to be sure that once this happens, the low information/process is trapped: it can neither carry high-level information out of the boundary nor move out of it.

The notion of security boundary allows us to formally express *Direct Information Leakage* in terms of boundary crossings, and to provide computational methods to verify absence of (direct) information flow, either through the verification of some syntactic properties [13] or by applying a nesting analysis [4,6] that extends the Nielsons' flow logic analysis approach [23].

In this paper, we go one step further, facing the issue of detecting *Indirect Information Leakage* as well. To the best of our knowledge, there is no result in literature concerning indirect information leakage detection in Mobile Ambients. The only current work we are aware of follows a different approach aiming at defining a type system that guarantees non-interference in Boxed Ambients [14].

We extend the language by directly including boundaries in the semantics. The advantages of this choice is that it guarantees the absence of direct information flow and provides an access control mechanism, still leaving open the much more intriguing issue of facing indirect information flow.

Then, to formally define (indirect) information leakage, we follow the standard approach based on non-interference [21,27]. Informally, there is no confidential data leakage if the system behaviour is not influenced by high level values/processes, i.e, if the high level part of the system is not able to "interfere" with the low level one. In other words, there is non-interference if no low-level observer can distinguish two processes that are equal modulo the values of confidential data. We formalize this idea through *contextual equivalence* [22,25], by requiring that the perturbation of high level values/processes does not change the observable behaviour of the system.

In order to check whenever a process is interference free, we extend the Nielsons' control flow analysis of [23] in two directions.

- First, the presence of boundary names in the language allows us to improve on the analysis, as constraints on boundary crossings reduce the occurrences of possible nestings.
- Second, a set of "sensitive" ambients is computed which contains all high ambients and is closed by nesting under the following condition: every ambient that may exercise a capability on a "sensitive ambient" is sensitive too.

Informally speaking, this set of sensitive ambients is an over approximation of ambients where perturbation propagates. We prove that if the set of "sensitive" ambients is protected inside security boundaries, then there is no information leakage towards any context that "well-behaves" with respect to the same set of sensitive ambients.

The main novelties of our approach with respect to previous attempts to tackle the problem of non-interference can be summarized as follows:

- The notion of boundary allows us to consider separately the problem of detecting direct and indirect information flows: the former can be expressed in terms of boundary crossing, while the latter can be formulated in terms of a localized propagation of confidential data perturbation;
- The control flow analysis approach allows us to infer an over approximation of ambient nestings and sensitive ambients. Thus, it leads to positive information also when process $P$ is not recognized as interference-free, as it may dramatically reduce the size of code inspection to find possible causes of information leakage. This is valuable when compared with the verification approach by prescriptive rules like in type-system approaches.

The rest of the paper is organized as follows. In Section 2, we introduce Boundary Ambient Calculus and we present the model of multilevel security for mobile agents. In Section 3 and 4, we deal with both direct and indirect information leakage, by specifying a suitable control flow analysis. Section 5 concludes the paper with final remarks and comparisons with related works.

## 2 The B-Ambients Calculus

The Mobile Ambients calculus has been introduced in [12] with the main purpose of explicitly modeling mobility. Indeed, ambients are arbitrarily nested boxes which can move around through suitable capabilities. *Boundary Ambients* (*B-Ambients*, for short) extend Mobile Ambients with special ambients, called *boundaries*, that are responsible of confining confidential information.

The syntax of processes is the same of Mobile Ambients, and it is given in Figure 1.

Notice that, differently from Mobile Ambients, here a name $n \in \mathsf{Names}$ denotes either an ambient or a boundary name. In particular, the countably infinite set of names $\mathsf{Names}$ is partitioned into two disjoint sets $\mathsf{Amb}$ and $\mathsf{Bound}$,

| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $(\nu n)P$ | restriction |
| | $\mid$ | $\mathbf{0}$ | inactivity |
| | $\mid$ | $P \mid Q$ | composition |
| | $\mid$ | $!P$ | replication |
| | $\mid$ | $n^{\ell^a}[\, P \,]$ | ambient or boundary |
| | $\mid$ | $\mathbf{in}^{\ell^t}\, n.P$ | capability to enter $n$ |
| | $\mid$ | $\mathbf{out}^{\ell^t}\, n.P$ | capability to exit $n$ |
| | $\mid$ | $\mathbf{open}^{\ell^t}\, n.P$ | capability to open $n$ |

Fig. 1. Boundary Ambients Syntax

with $\mathsf{Amb}$ representing the set of all ambient names and $\mathsf{Bound}$ representing the set of all boundary names. In the following, ambient names will range over $s$ and $t$, boundary names over $b$ and $d$, while $m$ and $n$ will denote both boundary and ambient names. Labels $\ell^a \in \mathbf{Lab}^a$ on ambients and labels $\ell^t \in \mathbf{Lab}^t$ on transitions are introduced, as it is customary in static analysis, to indicate "program points". They will be useful in Section 4 when developing the analysis, moreover ambient labelling is used to assign different security levels to ambients, in order to formalize a notion of information leakage.

Intuitively, the restriction $(\nu n)P$ introduces the new name $n$ and limits its scope to $P$; process $\mathbf{0}$ does nothing; $P \mid Q$ is $P$ and $Q$ running in parallel; replication provides recursion and iteration as $!P$ represents any number of copies of $P$ in parallel. By $n^{\ell^a}[\, P \,]$ we denote the ambient or boundary named $n$ with the process $P$ running inside it. The capabilities $\mathbf{in}^{\ell^t}\, n$ and $\mathbf{out}^{\ell^t}\, n$ move their enclosing ambients in and out ambient $n$, respectively; the capability $\mathbf{open}^{\ell^t}\, n$ is used to dissolve the boundary of a sibling ambient $n$. Moves over boundaries are controlled: $\mathbf{out}^{\ell^t}\, n$ and $\mathbf{open}^{\ell^t}\, n$, when $n$ is a boundary, are allowed only when the executing ambient is itself a boundary. The only ambient name binding operator is $\nu$: names that are not bound by a $\nu$ operator are thus free names. We denote by $fn(P)$ the set of free names of process P, and by $bn(P)$ the set of bound names, as usual.

The operational semantics of a process $P$ is given through a suitable reduction relation $\rightarrow$ and a structural congruence $\equiv$ between processes. They are depicted in Figures 3 and 2, respectively. Intuitively, $P \rightarrow Q$ represents the possibility for $P$ of reducing to $Q$ through some computation. Note that the reduction rules for the *open* and the *out* capabilities executed on a boundary require that such capabilities are performed by boundaries.

For the sake of readability, in all the following examples, we will omit the terminating $\mathbf{0}$ at the end of every process specification, e.g., we write $\mathbf{in}\, n \mid n[\,]$ in place of $\mathbf{in}\, n.\mathbf{0} \mid n[\,\mathbf{0}\,]$. In addition, we will use the notation $n^{\ell}[\![\, P \,]\!]$ to denote a boundary named $n$ and labelled $\ell$.

**Example 2.1** Let $P$ be a process modelling a $safe$ inside a bank $caveau$,

$P \equiv P$

$P \equiv Q \Rightarrow Q \equiv P$

$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$

$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q,\ n \in \mathsf{Names}$

$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$

$P \equiv Q \Rightarrow\ !P \equiv\ !Q$

$P \equiv Q \Rightarrow\ n^{\ell^a}[\,P\,] \equiv n^{\ell^a}[\,Q\,],\ n \in \mathsf{Names}$

$P \equiv Q \Rightarrow \mathbf{in}\, n.P \equiv \mathbf{in}\, n.P$

$P \equiv Q \Rightarrow \mathbf{out}\, n.P \equiv \mathbf{out}\, n.Q$

$P \equiv Q \Rightarrow \mathbf{open}\, n.P \equiv \mathbf{open}\, n.Q$

$P \mid Q \equiv Q \mid P$

$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$

$!P \equiv P \mid !P$

$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P,\ n, m \in \mathsf{Names}$

$(\nu n)P \mid Q \equiv P \mid (\nu n)Q$ if $n \notin fn(P) \wedge n \in \mathsf{Names}$

$(\nu n)\, m^{\ell^a}[\,P\,] \equiv m^{\ell^a}[\,(\nu n)P\,]$ if $n \neq m \wedge n, m \in \mathsf{Names}$

$P \mid \mathbf{0} \equiv P$

$(\nu n)\mathbf{0} \equiv \mathbf{0},\ n \in \mathsf{Names}$

$!\mathbf{0} \equiv \mathbf{0}$

Fig. 2. Structural Congruence: $P \equiv Q$

(InRed) $n^{\ell_1^a}[\,\mathbf{in}^{\ell^t}\, m.P \mid Q\,] \mid m^{\ell_2^a}[\,R\,] \rightarrow m^{\ell_2^a}[\, n^{\ell_1^a}[\,P \mid Q\,] \mid R\,]$ $m, n \in \mathsf{Names}$

(OutRed) $m^{\ell_1^a}[\, n^{\ell_2^a}[\,\mathbf{out}^{\ell^t}\ m.P \mid Q\,] \mid R\,] \longrightarrow n^{\ell_2^a}[\,P \mid Q\,] \mid m^{\ell_1^a}[\,R\,]$ $m, n \in \mathsf{Amb} \vee m, n \in \mathsf{Bound}$

$s^{\ell_1^a}[\, b^{\ell_2^a}[\,\mathbf{out}^{\ell^t}\ s.P \mid Q\,] \mid R\,] \longrightarrow b^{\ell_2^a}[\,P \mid Q\,] \mid s^{\ell_1^a}[\,R\,]$ $s \in \mathsf{Amb} \wedge b \in \mathsf{Bound}$

(OpenRed) $m^{\ell_1^a}[\,\mathbf{open}^{\ell^t}\ n.P \mid n^{\ell_2^a}[\,Q\,] \mid R\,] \longrightarrow m^{\ell_1^a}[\,P \mid Q \mid R\,]$ $m, n \in \mathsf{Amb} \vee m, n \in \mathsf{Bound}$

$b^{\ell_1^a}[\,\mathbf{open}^{\ell^t}\ s.P \mid s^{\ell_2^a}[\,Q\,] \mid R\,] \longrightarrow b^{\ell_1^a}[\,P \mid Q \mid R\,]$ $s \in \mathsf{Amb} \wedge b \in \mathsf{Bound}$

(ResRed) $P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$

(AmbRed) $P \rightarrow Q \Rightarrow\ n^{\ell^a}[\,P\,] \rightarrow n^{\ell^a}[\,Q\,]$

(CompRed) $P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$

(InRed) $P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$

Fig. 3. Reduction Rules: $P \rightarrow Q$

i.e., $P = caveau^{\ell_1^a}[\![\, safe^{\ell_2^a}[\![\, \mathbf{out}^{\ell^t}\, caveau\,]\!] \mid \mathbf{open}^{\ell^t}\, safe\,]\!] \mid \mathbf{open}^{\ell^t}\, safe$. Notice that, according to the semantics of B-Ambients, the $\mathbf{open}^{\ell^t}\, safe$ can be performed only when $safe$ is inside $caveau$, thus preventing $safe$ to be opened by any ambient at environment level.

*2.1 Observational Equivalences*

The notion of information leakage we will introduce in Section 3.2 is based on a Context Equivalence relation parameterized on a set of contexts. This relation equates processes that are indistinguishable by such a set of contexts with respect to the set of ambient/boundary names there may exist. Let us then recall from [22] some useful definitions about contexts and observables.

**Definition 2.2** (**Context:** $\mathcal{C}$) A context $\mathcal{C}$ is a process containing zero or more holes. In the following, we write $\mathcal{C}(P)$ for the outcome of filling each hole in the context $\mathcal{C}$ with process $P$.

For example, let P be a process of the form $P = s^{\ell_1^a}[\,Q\,] \mid b^{\ell_2^a}[\![\,R\,]\!]$, and $\mathcal{C}$ a context of the form $\mathcal{C}(\_) = t^{\ell_3^a}[\,S\,] \mid d^{\ell_4^a}[\![\,\_\,]\!]$. Thus, $\mathcal{C}(P) = t^{\ell_3^a}[\,S\,] \mid d^{\ell_4^a}[\![\; s^{\ell_1^a}[\,Q\,] \mid b^{\ell^a 2}[\![\,R\,]\!]\;]\!]$. Notice that names which are free in $P$ may become bound in $\mathcal{C}(P)$.

The following definitions mean intuitively that an observer may eventually detect an ambient named $n$ at the top-level of process $P$.

**Definition 2.3** (**Exhibition of a Name:** $P \downarrow n$) Let P be a process, and n $\in$ Names the name of either a boundary or an ambient. Then, P exhibits name n (P $\downarrow$ n) iff there are $m_1, m_2, ..., m_k$ with $m_i \neq n\ \forall i \leq k$, and two processes $P^{'}$ and $P^{''}$ such that $P \equiv (\nu m_1, m_2, ..., m_k)(\ n[\,P^{'}\,] \mid P^{''})$.

**Definition 2.4** (**Convergence to a Name:** $P \Downarrow n$) Let P be a process, and n $\in$ Names the name of either a boundary or an ambient. Then, P converges to name n (P $\Downarrow$ n) iff $\exists Q\ s.t.\ P \rightarrow^* Q \wedge Q \downarrow n$.

The next definition is a parameterized version of Contextual Equivalence [22].

**Definition 2.5** (**Contextual Equivalence up to** $\mathbb{C}$: $P \simeq_{\mathbb{C}} P^{'}$) Let $\mathbb{C}$ be a set of contexts. Two processes $P$ and $P^{'}$ are contextually equivalent up to $\mathbb{C}$, denoted $(P \simeq_{\mathbb{C}} P^{'})$, iff for all $n \in$ Names and $\mathcal{C} \in \mathbb{C}$, $\mathcal{C}(P) \Downarrow n \Leftrightarrow \mathcal{C}(P^{'}) \Downarrow n$.

In the technical proofs we will use the following definition of barbed bisimilarity and barbed congruence.

**Definition 2.6** (**Barbed bisimilarity:** $\mathbf{P \approx P^{'}}$) A barbed bisimulation parameterized to a set of contexts $\mathbb{C}$ is a symmetric relation $\mathcal{S}$ such that whenever $(P, P^{'}) \in \mathcal{S}$,

- $P \downarrow n$ implies $P^{'} \downarrow n$;
- $P \longrightarrow Q$ implies that $\exists Q'$ such that $P' \longrightarrow^* Q'$ and $(Q, Q^{'}) \in \mathcal{S}$.

Two processes $P$ and $P'$ are barbed bisimilar $(P \approx P^{'})$ iff there exists a barbed bisimulation $\mathcal{S}$ such that $(P, P') \in \mathcal{S}$.

**Definition 2.7** (**Barbed congruence up to** $\mathbb{C} : \mathbf{P \approx_{\mathbb{C}} P^{'}}$) Let $\mathbb{C}$ be a set of contexts. Two processes $P$ and $P^{'}$ are barbed congruent up to $\mathbb{C}$, denoted $(P \approx_{\mathbb{C}} P^{'})$, iff for all $\mathcal{C} \in \mathbb{C}$, $\mathcal{C}(P) \approx \mathcal{C}(P^{'})$.

The following result shows that, as expected, barbed congruence is stronger than contextual equivalence.

**Proposition 2.8** *Let $\mathbb{C}$ be a set of contexts. $P \approx_{\mathbb{C}} P'$ implies $P \simeq_{\mathbb{C}} P'$.*

***Proof.*** Assume $P \approx_{\mathbb{C}} P'$ and consider a context $\mathcal{C} \in \mathbb{C}$ and a name $n \in \mathsf{Names}$. We prove that $\mathcal{C}(P) \Downarrow n$ implies $\mathcal{C}(P^{'}) \Downarrow n$. $\mathcal{C}(P) \Downarrow n$ means that $\exists Q$ s.t. $\mathcal{C}(P) \rightarrow^* Q \wedge Q \downarrow n$. Since $P \approx_{\mathbb{C}} P'$ and $\mathcal{C} \in \mathbb{C}$ we also have $\mathcal{C}(P) \approx \mathcal{C}(P')$. By definition of barbed bisimilarity, $\mathcal{C}(P') \rightarrow^* Q'$, with $Q \approx Q'$, thus $Q \downarrow n$ implies $Q' \downarrow n$, hence proving that $\mathcal{C}(P') \Downarrow n$.

The fact that $\mathcal{C}(P') \Downarrow n$ implies $\mathcal{C}(P) \Downarrow n$ may be symmetrically proved. We thus have that for all $\mathcal{C} \in \mathbb{C}$ and for all names $n \in \mathsf{Names}$, $\mathcal{C}(P) \Downarrow n$ iff $\mathcal{C}(P^{'}) \Downarrow n$, and so the thesis $P \simeq_{\mathbb{C}} P'$. □

The result above allows to use barbed congruence as a sufficient condition for proving contextual equivalence.

### *2.2 Modeling Multilevel Security*

In order to define Multilevel security in B-Ambients, we first need to classify information into different levels of confidentiality. We do this by exploiting the labelling of ambients and boundaries. In particular, we partition the set of labels $\mathbf{Lab}^a$ into three disjoint sets $\mathbf{Lab}^a_H, \mathbf{Lab}^a_L$ and $\mathbf{Lab}^a_B$, which stand for *high*, *low* and *boundary* labels. We require that all (and only) boundaries must be labelled with boundary labels from set $\mathbf{Lab}^a_B$. *High level* ambients are the ones labelled in $\mathbf{Lab}^a_H$. We require that they are nested in boundaries. Finally, all the other ambients are considered *low level* ones and they are consequently labelled with labels from set $\mathbf{Lab}^a_L$. This is how we will always label processes, and it corresponds to defining a multilevel security policy: what is confidential (high), what is possibly malicious (low), what is a container of possible secrets (boundary). In all the examples, we will use the following notation for labels: $b \in \mathbf{Lab}^a_B, h \in \mathbf{Lab}^a_H$, $m, m^{'} \in \mathbf{Lab}^a_L$ and $c, ch, cl, cm, cm^{'} \in \mathbf{Lab}^t$.

As an example, consider the following process:

$$P_1 = container^b[\![\ hdata^h[\,\mathbf{out}^c\, container\,]\ \mid\ send^{b^{'}}[\![\,\mathbf{out}^{c^{'}}\, container.Q\,]\!]\ ]\!]$$

Ambient *container* is a boundary for high level data *hdata* (note that data are abstractly represented as ambients). This process is an example of how boundaries may prevent direct information flow. In fact, the high level ambient *hdata* cannot perform the *out* capability from boundary *container*, as it is prevented by the semantics of B-Ambients. Thus, *hdata* is never exposed to any ambient at environment level. [4] On the other hand, ambient *send*, which is a boundary, may go out of *container* without caus-

[4] Note that the presence of an ambient may be tested by trying to open it or by entering and then exiting from it. A low level ambient may thus test if *hdata* is present. This may be seen as reading high level information.

$$
\begin{array}{lll}
(res) & Nest_\ell((\nu n)P) & = Nest_\ell(P) \\
(zero) & Nest_\ell(\mathbf{0}) & = \emptyset \\
(par) & Nest_\ell(P \mid Q) & = Nest_\ell(P) \cup Nest_\ell(Q) \\
(repl) & Nest_\ell(!P) & = Nest_\ell(P) \\
(amb) & Nest_\ell(\, n^{\ell^a}[\, P \,]\,) & = Nest_\ell(P) \cup \{(\ell, \ell^a)\} \\
(in) & Nest_\ell(\mathbf{in}^{\ell^t}\, n.P) & = Nest_\ell(P) \cup \{(\ell, \ell^t)\} \\
(out) & Nest_\ell(\mathbf{out}^{\ell^t}\, n.P) & = Nest_\ell(P) \cup \{(\ell, \ell^t)\} \\
(open) & Nest_\ell(\mathbf{open}^{\ell^t}\, n.P) & = Nest_\ell(P) \cup \{(\ell, \ell^t)\}
\end{array}
$$

Fig. 4. Definition of Function $Nest$

ing any direct information leakage. In particular, $P_1$ may only evolve to $container^b[\![\ hdata^h[\,\mathbf{out}^c\, container\,]\,]\!] \mid send^{b'}[\![\, Q \,]\!]$.

## 3 Direct and Indirect Information Flow

### *3.1 Direct Information Flow*

The concept of direct flow of high level/ambients outside security boundaries may be formalized as follows. In the definitions we use the function $Nest$ reported in Figure 4 which collects all the nestings and capabilities of a given process $P$. We denote by $env$ a special label corresponding to the low level environment.

**Definition 3.1** (**Unprotected**) Given a process $P$, a labelling $\mathbf{Lab}^a$, and a set $R \subseteq (\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$, $\texttt{Unprotected}\ (\ell,R) = \text{true}$ iff $\exists\, \ell_1, ..., \ell_n \in \mathbf{Lab}^a_L$ s.t $(env, \ell_1), (\ell_1, \ell_2), ..., (\ell_{n-1}, \ell_n), (\ell_n, \ell) \in R$.

**Definition 3.2** (**Protected**) Given a process $P$, a labelling $\mathbf{Lab}^a$, and a set $R \subseteq (\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$, $\texttt{Protected}\ (\ell,R) = \text{true}$ iff $\neg\texttt{Unprotected}\ (\ell,R)$.

**Definition 3.3** (**Direct Information Leakage**) Given a process $P$, a labelling $\mathbf{Lab}^a$, and a set $R \subseteq (\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$, $P$ directly leaks secret h $\in \mathbf{Lab}^a_H$ iff $\exists Q, P \rightarrow^* Q$ such that $\texttt{Unprotected}\ (h, Nest_\ell(Q))$.

**Example 3.4** In distributed and mobile systems, it is unrealistic to consider a unique boundary, containing all the confidential information. As an example, consider *bob* and *alice* willing to exchange some confidential information that need to be protected during the communication process. This can be modeled by defining two boundaries, one for each principal: $bob^b[\![\, Q_1 \,]\!] \mid alice^b[\![\, Q_2 \,]\!] \mid Q$. Making the model applicable needs a mechanism for moving confidential data from one boundary to another one. This

may be achieved through another boundary which moves out from the first protected area and into the second one (i.e. a sort of encryption of the confidential data sent from one actor to the other). The following example, also depicted in Figure 5, describes the exchange of confidential information between the two principals *bob* and *alice*:

$$P_2 = bob^b[\![\ encrypt^b[\![\ \mathbf{out}^c\, bob.\mathbf{in}^c\, alice\ ]\!]\ |\ hdata^h[\ \mathbf{in}^{ch}\, encrypt\ ]\ ]\!]\ |\ alice^b[\![\ \mathbf{open}^c\, encrypt\ ]\!]\ |\ Q$$

The process may evolve to the following one (see steps (a) to (d) of Figure 5):

$$bob^b[\![\ ]\!]\ |\ alice^b[\![\ \mathbf{open}^c\, encrypt\ |\ encrypt^b[\![\ hdata^h[\ ]\ ]\!]\ ]\!]\ |\ Q$$

and finally to (see step (e) of Figure 5):

$$bob^b[\![\ ]\!]\ |\ alice^b[\![\ hdata^h[\ ]\ ]\!]\ |\ Q$$

Note that *encrypt* is labelled as a boundary. Thus, high level data *hdata* is always protected by boundary ambients, during the whole execution. Furthermore, notice that an empty message could be sent from *bob* to *alice*. Such a situation could be avoided by making the model deterministic, but this would complicate the example.

### *3.2 Detecting Indirect Flows*

In this section, we face the issue of detecting indirect information flows. To do so, we follow the standard approach based on non-interference [21]. Informally, there is no information flow from high to low if and only if the system behaviour is not influenced by high level values/processes, i.e, if and only if the high level part of the system is not able to interfere with the low level one (see, .e.g, [17,18,20] for more detail on non-interference-based properties). We formalize this idea by requiring that the perturbation of high level values/processes do not change the observable behaviour of the system. If this happens, we should be guaranteed that no interference is possible from level high to low.

**Example 3.5** Let $P_3$ be the following process:

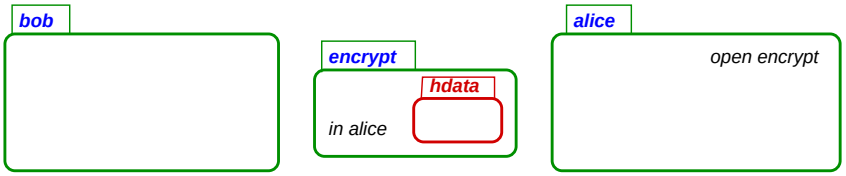
$$container^b[\![\ send^b[\![\ \mathbf{in}^c\, hdata.\mathbf{out}^c\, hdata.\mathbf{out}^c\, container\ ]\!]\ |\ \mathbf{open}^c\, download\ ]\!]$$

In this process, the fact that *send* exits from *container* is caused by the presence of *hdata*, e.g. in a downloaded application. There is no direct flow, but a low level user deduces information about high level ambients in an indirect way.

**Definition 3.6** (**Substitution Function** $\sigma_N$) Let $N \in \mathsf{Names}$ be a set of names. A substitution function $\sigma_N$ over $N$ is a function $\sigma_N : \mathsf{Names} \to \mathsf{Names}$, such that $\sigma_N(s) = s$, whenever $s \notin N$. We also denote with $P\sigma_N$ the process $P$ in which $\sigma_N$ is applied to all the name occurrences.


(a) Bob needs to send confidential data "hdata" to Alice.


(b) The confidential data is encrypted with a shared key.




(c) The confidential data is sent encrypted over the communication channel.


(d) The confidential data is safely received by alice.


(e) Alice accesses the confidential data by decrypting it.

Fig. 5. Bob and Alice exchange confidential information

We now give a formal definition of absence of information leakage based on non-interference by adopting the approach introduced and discussed in [1].

**Definition 3.7** (**Absence of (Indirect) Information Leakage**) Let $P$ be a process, $N \subseteq \mathsf{Names}$ be a set of names and $\mathbb{C}$ be a set of contexts. P does not leak secrets $N$ to $\mathbb{C}$ if and only if, for all substitution functions $\sigma_N$ we have $P \simeq_{\mathbb{C}} P\sigma_N$

Intuitively, $N$ represents the set of sensitive names, corresponding to ambients that belong to the propagation area when perturbations of confidential data may arise. Given a set of contexts $\mathbb{C}$, we say that $P$ does not leak high level information to $\mathbb{C}$ if and only if any perturbation of such information is not visible whenever $P$ is executed in every context $\mathcal{C} \in \mathbb{C}$.

**Example 3.8** Consider again process $P_3$ above. We show that it leaks name

*hdata*. Consider context $\mathcal{C}(\_) = \_ \mid \mathit{download}^b[\![\, \mathbf{in}^c\, \mathit{container} \mid \mathit{hdata}^h[\,\mathbf{0}\,]\,]\!]$, and substitution $\sigma_{\mathit{hdata}}(\mathit{hdata}) = \mathit{hdata}'$, $\sigma_{\mathit{hdata}}(n) = n\ \forall n \neq \mathit{hdata}$. Then, $\mathcal{C}(P) \Downarrow \mathit{send}$ while $\mathcal{C}(P\sigma_{\mathit{hdata}}) \not\Downarrow \mathit{send}$. Therefore, $P \not\approx_{\{\mathcal{C}\}} P\sigma_{\mathit{hdata}}$, i.e., $P$ leaks *hdata* to $\mathcal{C}(\_)$.

In this process, *send* exits from *container* by the presence of *hdata*, e.g. in a downloaded application. There is no direct flow, but a low level user might deduce information about high level ambients in an indirect way.

Our aim is now to extend the techniques of [4,6] for direct flows, to handle also indirect information leakage. To this purpose, we define a suitable control flow analysis that allows us to statically verify that a system $P$ do not leaks secrets.

## 4 Control Flow Analysis of Information Leakage

The control flow analysis we propose aims at modeling which ambients may be influenced by high level ambients, i.e., which is the part of the system whose execution is influenced by the presence of high level information. It works on triplets $(\hat{C},\hat{I},\hat{H})$, where $\hat{I}$ and $\hat{H}$ are used to track the ambient nestings at runtime as done in [23], and $\hat{C}$ represents the set of ambients whose execution is sensitive. More specifically:

- The first component $\hat{C}$ (sensitive ambients) is an element of $\wp(\mathsf{Names})$. If a process contains an ambient $n$ whose execution is conditioned by high level ambients, then $n$ should be in $\hat{C}$.
- The second component $\hat{I}$ (ambients nestings) is an element of $\wp(\mathbf{Lab}^a \times (\mathbf{Lab}^a \cup \mathbf{Lab}^t))$. If a process contains an ambient labelled $\ell^a$ having inside either a capability or an ambient labelled $\ell$, then $(\ell^a, \ell)$ is expected to belong to $\hat{I}$.
- The third component $\hat{H} \in \wp(\mathbf{Lab}^a \times \mathsf{Names})$ keeps track of the correspondence between names and labels. If a process contains an ambient labelled $\ell^a$ with name $n$, then $(\ell^a, n)$ is expected to belong to $\hat{H}$.
- The triplets are component-wise partially ordered, and $\sqsubseteq$ is the component-wise inclusion.

According to the control flow framework in [26], the analysis is defined by a representation function and an analysis specification [5]. They are depicted, respectively, in Figure 6 and Figure 7.

The representation function aims at mapping concrete values to their best abstract representation. It is given in terms of a function $\beta^{\mathrm{B}}_{\ell}(P)$ which basically builds sets $\hat{C}$, $\hat{I}$ and $\hat{H}$ corresponding to process $P$, with respect to an enclosing ambient labelled with $\ell$. The representation of a process $P$ is

[5] In ambient calculus bound names may be $\alpha$-converted. For the sake of simplicity, here we are assuming that ambient names are *stable*, i.e., $n$ is indeed a representative for a class of $\alpha$-convertible names. See [23] for more details on how this can be handled.

defined as $\beta^{\mathrm{B}}_{env}(P)$. Intuitively, $\beta^{\mathrm{B}}_{env}(P)$ collects in $\hat{C}$ the names of ambients nested inside a high level ambient, and in $\hat{I}$ all the nestings of ambients and capabilities. Finally, in $\hat{H}$, $\beta^{\mathrm{B}}_{env}(P)$ collects all the mappings between labels and ambients.

**Example 4.1** Consider again process $P_1$ introduced in Section 2.2.

$$P_1 = \mathit{container}^b[\![\ \mathit{hdata}^h[\,\mathbf{out}^c\,\mathit{container}\,]\ \mid\ \mathit{send}^{b'}[\![\,\mathbf{out}^{c'}\,\mathit{container}.Q\,]\!]\ ]\!]$$

The representation function of P is the following: $\beta^{\mathrm{B}}_{env}(P) = (\{\mathit{hdata}\}, \{(\mathit{env}, b), (b, h), (b, b'), (h, c), (b', c')\}, \{(b, \mathit{container}), (h, \mathit{hdata}), (b', \mathit{send})\})$.

Let us briefly discuss the specification rules of Figure 7. They depict how the process transforms one abstract representation to another one, and they are logically divided in two parts: $(i)$ the construction of sets $\hat{I}$ and $\hat{H}$, which is done by refining the analysis of [23] in order to correctly handle boundaries, and $(ii)$ the construction of set $\hat{C}$ of sensitive ambients. The specification mostly amounts to recursive check of subprocesses except for the three capabilities *open*, *in*, and *out*, and for ambient/boundary definition. The first part of the rule for *open*-capability says that if ambient labelled $\ell^a$ has an *open*-capability $\ell^t$ on ambient $n$, that may apply due to the presence of a sibling ambient labelled $\ell^{a'}$ whose name is $n$, and either $\ell^{a'}$ is not a boundary or $\ell^a$ is a boundary, then the result of performing that capability should also be recorded in $\hat{I}$, i.e. all the ambients/capabilities nested in $\ell^{a'}$ have to be nested also in $\ell^a$. Notice that, requiring that either $\ell^{a'}$ is not a boundary or $\ell^a$ is a boundary, allows us to consider only the open capabilities that may indeed be performed (recall that only boundaries may open boundaries). The second part of the rule aims at collecting information about sensitive ambients/boundaries. It requires that every ambient/boundary potentially performing an output capability on a sensitive ambient/boundary, has to be considered sensitive, as well. The *in* and *out* capabilities behave similarly. Finally, the rule for ambient/boundary definition requires that every ambient/boundary nested in a sensitive ambient/boundary has to be considered sensitive, too.

**Example 4.2** Let $P_1$ be the process of Example 4.1. It is easy to see that the least solution of the analysis for $P_1$ is the triplet $(\hat{C}, \hat{I}, \hat{H})$ where $\hat{C} = \{\mathit{hdata}\}$, $\hat{I} = \{(\mathit{env}, b), (\mathit{env}, b'), (b, h), (b, b'), (h, c), (l, c')\}$, and $\hat{H} = \{(b, \mathit{container})$ $(h, \mathit{hdata}), (b', \mathit{send})\})$. Notice that, with respect to $\beta^{\mathrm{B}}_{env}(P_1)$, the analysis adds in $\hat{I}$ the pair $(\mathit{env}, l)$, representing the possibility for boundary *send* to exit the container. As there is no ambient/boundary performing capabilities over *hdata*, the set $\hat{C}$ of sensitive ambients/boundaries is not incremented, and contains *hdata* only.

The correctness of the analysis is proven by showing that every reduction of the semantics is properly mimicked in the analysis:

**Theorem 4.3** *Let $P$ and $Q$ be two processes such that $\beta^{\mathrm{B}}_{env}(P) \sqsubseteq (\hat{C}, \hat{I}, \hat{H}) \ \wedge$ $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P \ \wedge\ P \to Q$. Then, $\beta^{\mathrm{B}}_{env}(Q) \sqsubseteq (\hat{C}, \hat{I}, \hat{H}) \ \wedge\ (\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} Q$.*

***Proof.*** The correctness for sets $\hat{I}$ and $\hat{H}$ may be obtained by slightly adapting the proof for the control flow analysis proposed in [23]. Then, it is sufficient to observe that the *amb* rule requires that any ambient/boundary potentially nested, at run-time, inside a high level ambient, is added in $\hat{C}$. This proves that the $\hat{C}'$ produced by $\beta^{\mathrm{B}}_{env}(Q)$ is a subset of $\hat{C}$. □

Intuitively, the theorem above states that whenever $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$ and the representation of $P$ is contained in $(\hat{C}, \hat{I}, \hat{H})$, we are assured that every ambient which is nested inside a high level ambient and every nesting of ambients and capabilities in every possible derivative of $P$ is also captured in $(\hat{C}, \hat{I}, \hat{H})$. It is important to recall that the resulting control flow analysis applies to any process. It is also possible to prove that every process enjoys a *least* analysis.

We now state the main result of the paper by showing how the control flow defined above may be applied to prove the absence of (indirect) information flow in a process $P$.

First, some additional definitions need to be introduced. Given an analysis $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$, we write $\texttt{Protected}\ (\hat{C}, (\hat{I}, \hat{H}))$ to denote that for every $n \in \hat{C}$ such that $\exists (l, n) \in \hat{H}$, it holds $\texttt{Protected}\ (l, \hat{I})$. Moreover, we denote by *Cont* the set of contexts whose hole is not contained inside any high level ambient. In other words, a context in *Cont* cannot host a process inside a sensitive area. Finally, given $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$ we define

$$\begin{aligned}\mathbb{C}_{P,\hat{C}} = \{\mathcal{C} \in \mathit{Cont} \mid\ & bn(\mathcal{C}(\mathbf{0})) \cap fn(P) = \emptyset \wedge \exists (\hat{C}', \hat{I}', \hat{H}') \text{ s.t.} \\ & (\hat{C}', \hat{I}', \hat{H}') \models^{\mathrm{B}} \mathcal{C}(\mathbf{0}) \text{ and } \texttt{Protected}\ (\hat{C}', (\hat{I}', \hat{H}')) \wedge \\ & fn(P) \cap fn(\mathcal{C}(\mathbf{0})) \cap \hat{C} = fn(P) \cap fn(\mathcal{C}(\mathbf{0})) \cap \hat{C}'\}\end{aligned}$$

Intuitively, $\mathbb{C}_{P,\hat{C}}$ is the set of contexts $\mathcal{C} \in \mathit{Cont}$ that admit an analysis $(\hat{C}', \hat{I}', \hat{H}')$ such that $\hat{C}$ and $\hat{C}'$ classify the free-names of both $P$ and $\mathcal{C}(\mathbf{0})$, with respect to sensitivity, in the same way. The condition $bn(\mathcal{C}(\mathbf{0})) \cap fn(P) = \emptyset$ is necessary to avoid that the context binds the free names of $P$. Observe that such a condition is not restrictive, as it may be guaranteed through a $\alpha$-conversion of bound names.

**Lemma 4.4** *Let $P$ be a process and consider a triplet $(\hat{C}, \hat{I}, \hat{H})$ such that $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$ and* $\texttt{Protected}(\hat{C}, (\hat{I}, \hat{H}))$. *Then, for all $\mathcal{C} \in \mathbb{C}_{P,\hat{C}}$, there exists a triplet $(\hat{C}'', \hat{I}'', \hat{H}'')$ such that $(\hat{C}'', \hat{I}'', \hat{H}'') \models^{\mathrm{B}} \mathcal{C}(P)$ and* $\texttt{Protected}(\hat{C}'', (\hat{I}'', \hat{H}''))$.

***Proof.*** Consider an analysis $(\hat{C}, \hat{I}, \hat{H})$ for $P$ such that $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$ and such that $\texttt{Protected}(\hat{C}, (\hat{I}, \hat{H}))$. By definition of $\mathbb{C}_{P,\hat{C}}$, there exists an analysis $(\hat{C}', \hat{I}', \hat{H}') \models^{\mathrm{B}} \mathcal{C}(\mathbf{0})$ such that $\texttt{Protected}(\hat{C}', (\hat{I}', \hat{H}'))$. Now, if we get $\hat{C}'' = \hat{C} \cup \hat{C}'$, $\hat{H}'' = \hat{H} \cup \hat{H}'$, it is possible to prove that exists $\hat{I}'' \supseteq \hat{I} \cup \hat{I}'$ such that $(\hat{C}'', \hat{I}'', \hat{H}'') \models^{\mathrm{B}} \mathcal{C}(P)$ and $\texttt{Protected}(\hat{C}'', (\hat{I}'', \hat{H}''))$.

The fact that $\hat{H}'' = \hat{H} \cup \hat{H}'$ is trivially derived by how $\hat{H}$ is constructed, i.e., it is a collection of mappings between labels and ambients. The crucial

point to prove is that $\hat{C}'' = \hat{C} \cup \hat{C}'$ even if the nestings in $\hat{I}''$ may be more than in $\hat{I} \cup \hat{I}'$, because of the potential interaction between the process and the context. To this aim, we consider the least set $\hat{I}'' \supseteq \hat{I} \cup \hat{I}'$ such that $(\mathsf{Names}, \hat{I}'', \hat{H}'') \models^{\mathrm{B}} \mathcal{C}(P)$. We show that also $(\hat{C}'', \hat{I}'', \hat{H}'') \models^{\mathrm{B}} \mathcal{C}(P)$. It is important to observe that $fn(P) \cap fn(\mathcal{C}(\mathbf{0})) \cap \hat{C} = fn(P) \cap fn(\mathcal{C}(\mathbf{0})) \cap \hat{C}'$, i.e., that $P$ and the context agree on what is sensitive and what is not on the common free-names. The fact that $(\hat{C}'', \hat{I}'', \hat{H}'') \models^{\mathrm{B}} \mathcal{C}(P)$ is based on observing that a new sensitive ambient/boundary would be added by the analysis to $\hat{C} \cup \hat{C}'$ in the following two cases:

- when a non-sensitive ambient/boundary become nested in a sensitive one;
- when a non-sensitive ambient/boundary gains a new capability with a sensitive ambient/boundary as target.

Thanks to the minimality of $\hat{I}''$, the first case may only happen when a non-sensitive ambient/boundary enters a sensitive one (recall that the hole in the context is required to be a non-sensitive place). However the *in* rule guarantees that any potential *in* inside sensitive ambient/boundaries makes the performing ambient/boundary a sensitive one.

The second case may happen only through an *open* capability, as it is the only rule that adds new nestings of capabilities. However, notice that if the opened ambient contains an *open* capability over a sensitive ambient/boundary, then, by the *open* rule it must be sensitive, too. We thus obtain a non-sensitive ambient opening a sensitive one, again proving that the opening ambient/boundary is sensitive, too. □

We can now state the main theorem:

**Theorem 4.5** *Let $P$ be a process and consider a triplet $(\hat{C}, \hat{I}, \hat{H})$ such that $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P$ and $\mathtt{Protected}(\hat{C}, (\hat{I}, \hat{H}))$. Then, $P$ does not leak $\hat{C}$ to $\mathbb{C}_{P,\hat{C}}$.*

***Proof.*** We have to prove that for all substitution functions $\sigma_{\hat{C}}$ we have $P \simeq_{\mathbb{C}_{P,\hat{C}}} P\sigma_{\hat{C}}$, i.e., that $P$ does not leak $\hat{C}$ to $\mathbb{C}_{P,\hat{C}}$. To do this, we exploit Proposition 2.8 and we thus prove $P \approx_{\mathbb{C}_{P,\hat{C}}} P\sigma_{\hat{C}}$. This amounts to proving that for all contexts $\mathcal{C} \in \mathbb{C}_{P,\hat{C}}$ we have $\mathcal{C}(P) \approx \mathcal{C}(P\sigma_{\hat{C}})$.

We define a barbed bisimilarity as follows. Consider a transformation on processes $ns(P, \hat{C})$ that returns process $P$ where all ambients/boundaries belonging to $\hat{C}$, together with the process they include, have been syntactically replaced by $\mathbf{0}$ in $P$.

Now we define the following relation:

$$\mathcal{S} = \{\, (P, P') \mid \exists (\hat{C}, \hat{I}, \hat{H}), (\hat{C}', \hat{I}', \hat{H}') \text{ s.t. } (\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} P, (\hat{C}', \hat{I}', \hat{H}') \models^{\mathrm{B}} P', \\ \mathtt{Protected}(\hat{C}, (\hat{I}, \hat{H})), \mathtt{Protected}(\hat{C}', (\hat{I}', \hat{H}')), ns(P, \hat{C}) = ns(P', \hat{C}') \}$$

Intuitively, relation $\mathcal{S}$ equates processes that are the same with respect to non-sensitive ambients/boundaries and in which sensitive ambients are protected.

It is easy to prove that $\mathcal{S}$ is a barbed bisimilarity. The following steps of the proof can be summarized as follows.

(i) The fact that sensitive ambients are protected and $ns(P, \hat{C}) = ns(P', \hat{C}')$ proves that $P$ and $P'$ exhibit the same names (a protected ambient cannot float at top level).

(ii) By the definition of the control flow analysis, all the capabilities performed by non-sensitive ambients must have a non-sensitive ambient as target. Thus, since $ns(P, \hat{C}) = ns(P', \hat{C}')$, all such capabilities may be mutually simulated by $P$ and $P'$, moving to processes $Q$ and $Q'$, respectively, that still satisfy $ns(Q, \hat{C}) = ns(Q', \hat{C}')$ and $(\hat{C}, \hat{I}, \hat{H}) \models^{\mathrm{B}} \mathcal{C}(P), (\hat{C}', \hat{I}', \hat{H}') \models^{\mathrm{B}} \mathcal{C}(P'), \mathtt{Protected}(\hat{C}, (\hat{I}, \hat{H})), \mathtt{Protected}(\hat{C}', (\hat{I}', \hat{H}'))$. Note that, by Theorem 4.3, the control flow is invariant with respect to the execution of processes.

(iii) Finally, when a sensitive ambient/boundary performs a capability we may ignore such a move (no need to simulate it). This can easily proved if the target of the capability is a sensitive ambient/boundary, as $ns(P, \hat{C})$ is unchanged. The crucial case is when such capabilities are performed on non-sensitive ambients, as this might change the non-sensitive part of processes. Indeed, it is easy to see that *in* and *out* capabilities do not change the non-sensitive part of the process as they just move a non-sensitive part of it in a different place ($ns(P)$ is still the same). The most interesting case is the *open* capability. This could potentially modify the non-sensitive part of $P$. However, notice that the control flow rule *amb* requires that ambients/boundaries inside sensitive ambients/boundaries are also sensitive. This implies that whenever an *open* is performed by a sensitive ambient/boundary, the target should be inside such ambient/boundary, thus being sensitive. This basically proves that sensitive ambients/boundaries may perform *open* capabilities only over sensitive ambients/boundaries.

Once proved that $\mathcal{S}$ is a barbed bisimilarity, by Lemma 4.4 we have that there exists a triplet $(\hat{C} \cup \hat{C}', \hat{I}'', \hat{H} \cup \hat{H}')$ such that $(\hat{C} \cup \hat{C}', \hat{I}'', \hat{H} \cup \hat{H}') \models^{\mathrm{B}} \mathcal{C}(P)$ and $\mathtt{Protected}(\hat{C} \cup \hat{C}', (\hat{I}'', \hat{H} \cup \hat{H}'))$. Then, it is easy to see that $(\hat{C}\sigma_{\hat{C}} \cup \hat{C}', \hat{I}'', \hat{H}\sigma_{\hat{C}} \cup \hat{H}') \models^{\mathrm{B}} \mathcal{C}(P\sigma_{\hat{C}})$ and $\mathtt{Protected}(\hat{C}\sigma_{\hat{C}} \cup C', (\hat{I}'', \hat{H}\sigma_{\hat{C}} \cup \hat{H}'))$ and $ns(\mathcal{C}(P), \hat{C}\sigma_{\hat{C}} \cup \hat{C}') = ns(\mathcal{C}(P\sigma_{\hat{C}}), \hat{C}\sigma_{\hat{C}} \cup \hat{C}')$. Therefore $(\mathcal{C}(P), \mathcal{C}(P\sigma_{\hat{C}})) \in \mathcal{S}$, i.e. they are barbed bisimilar. □

**Example 4.6** The least analysis for process $P_1$ returns $\hat{C} = \{hdata\}$ which is also protected. Thus, $P_1$ does not leak $\hat{C}$. The same holds for process $P_2$. Regarding $P_3$, we have that the sensitive set $\hat{C}$ must contain *send*, as it performs capabilities over high level ambients. As *send* may exit from *container*, process $P_3$ cannot be proved interference-free, and indeed it is not.

Notice that in [19] also deadlocks are shown to be potential sources of information leakage. In order to deal with this special kind of information

flow, it is sufficient to strength Definition 3.7 by using an equivalence notion which is deadlock-sensitive, such as bisimilarity. Observe that Theorem 4.5 still holds even with this stronger notion of information leakage.

## 5 Related Works and Conclusions

The main novelty of the approach presented in this paper is that we face the problem of detecting indirect information leakage (non-interference) in the context of Mobile Ambients.

The most related contributions in the area mainly focused on either extending the ambient calculus thus enhancing its expressive power, or on building suitable type systems to verify security properties.

Among the type systems approaches, it is worth to mention [11], where the authors introduce a new type system for tracking the behaviour of mobile computations. Using groups, the type system can impose to an ambient behavioral constraints on the set of ambients it may cross and the set of ambients it may open. It has the effect of statically preventing certain communications through a mandatory access control policy, and can block accidental or malicious leakage of secrets. Dezani and Salvo [16] extend the work of Cardelli et al. just mentioned, with a type system that also expresses security levels associated with ambients and provide further control over ambient movement and opening.

Among the language extensions, some valuable proposals are described in [9,8,10,15,25]. *Safe Ambients* is a modification of Mobile Ambients, where a movement or an ambient dissolution can take place only when the affected ambient agrees, offering the corresponding coaction. *Boxed Ambients* [10] is another variant of the Ambient calculus with a completely different model of communication, which results from dropping the *open* capability. In their paper, Bugliesi et al. define also a type system that provides an effective mechanism for resource protection and access control. In [24], Degano et al. present a control flow analysis that mainly focuses on access control. The analysis relies on the use of coactions as a filter to control access to resources.

As already said in the Introduction, as far as we know the only work towards the study of non-interference in the context of Mobile Ambients is [14], where a type system that guarantees that well-typed programs do not interfere when in parallel with any high-level source is studied for Boxed Ambients. Of course, one of the priorities as future work will be to carefully compare these two approaches. This may also give interesting insights on the tradeoff between accuracy and efficiency and usability between type system and control-flow analysis techniques.

Finally, the way we propose to express a multilevel information flow policy through the notion of boundary, also leads to interesting applications to model cryptographic protocols [5].

## References


[1] Abadi, M. and A. D. Gordon, *A Bisimulation Method for Cryptographic Protocols*, Nordic Journal of Computing **5** (1998), pp. 267–303.

[2] Bell, D. E. and L. J. L. Padula, *"Secure Computer Systems: Unified Exposition and Multics Interpretation"*, ESD-TR-75-306, MITRE MTR-2997 (1976).

[3] Braghin, C., A. Cortesi, S. Filippone, R. Focardi, F. L. Luccio and C. Piazza, Banana*, a Tool for Boundary Ambients Nesting ANAlysis*, in: *Proc. of The 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, 2003.

[4] Braghin, C., A. Cortesi and R. Focardi, *Control Flow Analysis of Mobile Ambients with Security Boundaries*, in: B. Jacobs and A. Rensink, editors, *Proc. of Fifth IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'02)* (2002), pp. 197–212.

[5] Braghin, C., A. Cortesi and R. Focardi, *Freshness Analysis in Security Protocols*, in: *Proc. of 14th Nordic Workshop on Programming Theory (NWPT'02)*, 2002, pp. 30–33.

[6] Braghin, C., A. Cortesi and R. Focardi, *Security Boundaries in Mobile Ambients*, Computer Languages **28** (2002), pp. 101–127.

[7] Braghin, C., A. Cortesi, R. Focardi and S. van Bakel, *Boundary Inference for Enforcing Security Policies in Mobile Ambients*, in: *Proc. of The 2nd IFIP International Conference on Theoretical Computer Science (TCS'02)* (2002), pp. 383–395.

[8] Bugliesi, M. and G. Castagna, *Secure Safe Ambients*, in: *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)* (2001), pp. 222–235.

[9] Bugliesi, M. and G. Castagna, *Behavioural Typing of Safe Ambients*, Computer Languages **28** (2002), pp. 61–99, revised and extended version of [8].

[10] Bugliesi, M., G. Castagna and S. Crafa, *Boxed Ambients*, in: *Proc. of the 4th Int. Conference on Theoretical Aspects of Computer Science (TACS'01)*, LNCS **2215** (2001), pp. 38–63.

[11] Cardelli, L., G. Ghelli and A. D. Gordon, *Ambient Groups and Mobility Types*, in: J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses and T. Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, Proceedings of the International IFIP Conference TCS 2000 (Sendai, Japan)*, LNCS **1872**, IFIP (2000), pp. 333–347.

[12] Cardelli, L. and A. D. Gordon, *Mobile Ambients*, in: M. Nivat, editor, *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS)*, LNCS **1378**, Springer-Verlag, Berlin, Germany, 1998 pp. 140–155.

[13] Cortesi, A. and R. Focardi, *Information Flow Security in Mobile Ambients*, in: *Proc. of International Workshop on Concurrency and Coordination (ConCoord'01)*, Electronic Notes on Theoretical Computer Science, ENTCS **54** (2001).

[14] Crafa, S., M. Bugliesi and G. Castagna, *Information Flow Security for Boxed Ambients*, in: *F-WAN: Int. Workshop on Foundations of Wide Area Networks*, number 66(3) in ENTCS, 2002.

[15] Degano, P., F. Levi and C. Bodei, *Safe Ambients: Control Flow Analysis and Security*, in: J. He and M. Sato, editors, *Proc. of Advances in Computing Science - ASIAN'00 (6th Asian Computing Science Conference, Penang, Malaysia)*, LNCS **1961** (2000), pp. 199–214.

[16] Dezani-Ciancaglini, M. and I.Salvo, *Security Types for Mobile Safe Ambients*, in: J. He and M. Sato, editors, *Proc. of Advances in Computing Science - ASIAN'00 (6th Asian Computing Science Conference, Penang, Malaysia)*, LNCS **1961** (2000), pp. 215–236.

[17] Focardi, R. and R. Gorrieri, *A Classification of Security Properties for Process Algebras*, Journal of Computer Security **3** (1995), pp. 5–33.

[18] Focardi, R. and R. Gorrieri, *The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties*, IEEE Transactions on Software Engineering **23** (1997), pp. 550–571.

[19] Focardi, R. and R. Gorrieri, *Classification of security properties (part i: Information flow)*, in: *Foundations of Security Analysis and Design - Tutorial Lectures*, LNCS **2171** (2001), pp. 331–396.

[20] Focardi, R., R. Gorrieri and F. Martinelli, *Information Flow Analysis in a Discrete-Time Process Algebra*, in: *Proc. of The 13th Computer Security Foundations Workshop (CSFW)* (2000), pp. 170–184.

[21] Goguen, J. and J. Meseguer, *Security Policies and Security Models*, in: *Proc. of Symposium on Security and Privacy* (1992), pp. 11–20.

[22] Gordon, A. D. and L. Cardelli, *Equational Properties of Mobile Ambients*, in: W. Thomas, editor, *Proc. of the Second International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '99), Held as Part of the Joint European Conferences on Theory and Practice of Software (ETAPS'99), (Amsterdam, The Netherlands, April 1999)*, LNCS **1578** (1999), pp. 212–226.

[23] Hansen, R. R., J. G. Jensen, F. Nielson and H. R. Nielson, *Abstract Interpretation of Mobile Ambients*, in: A.Cortesi and G.File', editors, *Proc. of Static Analysis Symposium (SAS'99)*, number 1694 in Lecture Notes in Computer Science (1999), pp. 134–148.

[24] Levi, F. and C. Bodei, *Security Analysis for Mobile Ambients*, in: *Electronic Proc. of IFIP WG 1.7 Workshop on Issues on the Theory of Security (WITS'00), Geneve*, 2000.

[25] Levi, F. and D. Sangiorgi, *Controlling Interference in Ambients*, in: *Proc. 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, 2000, pp. 352–364.

[26] Nielson, F., H. R. Nielson and C. L. Hankin, "Principles of Program Analysis," Springer, 1999.

[27] Smith, G. and D. Volpano, *Secure Information Flow in a Multi-Threaded Imperative Language*, in: *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, New York, NY, 1998, pp. 355–364.

$$
\begin{array}{lll}
 & \beta^{\mathrm{B}}(P) & = \beta^{\mathrm{B}}_{env,False}(P) \\
(res) & \beta^{\mathrm{B}}_{\ell,Cond}((\nu n)P) & = \beta^{\mathrm{B}}_{\ell,Cond}(P) \\
(zero) & \beta^{\mathrm{B}}_{\ell,Cond}(\mathbf{0}) & = (\emptyset,\emptyset,\emptyset) \\
(par) & \beta^{\mathrm{B}}_{\ell,Cond}(P \mid Q) & = \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup \beta^{\mathrm{B}}_{\ell,Cond}(Q) \\
(repl) & \beta^{\mathrm{B}}_{\ell,Cond}(!P) & = \beta^{\mathrm{B}}_{\ell,Cond}(P) \\
(amb) & \beta^{\mathrm{B}}_{\ell,Cond}(\, n^{\ell^a}[\![\, P \,]\!]\,) & = \text{case } \mathit{Cond} \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell^a,Cond}(P) \sqcup (\{n\},\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
 & & \quad \text{False: if } (\ell_a \in \mathbf{Lab}^a_H) \text{ then} \\
 & & \qquad \beta^{\mathrm{B}}_{\ell^a,True}(P) \sqcup (\{n\},\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
 & & \quad \text{else} \\
 & & \qquad \beta^{\mathrm{B}}_{\ell^a,False}(P) \sqcup (\emptyset,\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
(bound) & \beta^{\mathrm{B}}_{\ell,Cond}(\, n^{\ell^a}[\![\, P \,]\!]\,) & = \text{case } \mathit{Cond} \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell^a,Cond}(P) \sqcup (\{n\},\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
 & & \quad \text{False: if } (\ell_a \in \mathbf{Lab}^a_H) \text{ then} \\
 & & \qquad \beta^{\mathrm{B}}_{\ell^a,True}(P) \sqcup (\{n\},\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
 & & \quad \text{else} \\
 & & \qquad \beta^{\mathrm{B}}_{\ell^a,False}(P) \sqcup (\emptyset,\{(\ell,\ell^a)\},\{(\ell^a,n)\}) \\
(in) & \beta^{\mathrm{B}}_{\ell,Cond}(\mathbf{in}^{\ell^t}\, n.P) & = \text{case } \mathit{Cond} \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\{n\},\{(\ell,\ell^t)\},\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset) \\
(out) & \beta^{\mathrm{B}}_{\ell,Cond}(\mathbf{out}^{\ell^t}\, n.P) & = \text{case } \mathit{Cond} \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\{n\},\{(\ell,\ell^t)\},\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset) \\
(open) & \beta^{\mathrm{B}}_{\ell,Cond}(\mathbf{open}^{\ell^t}\, n.P) & = \text{case } \mathit{Cond} \text{ of} \\
 & & \quad \text{True} : \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\{n\},\{(\ell,\ell^t)\},\emptyset) \\
 & & \quad \text{False: } \beta^{\mathrm{B}}_{\ell,Cond}(P) \sqcup (\emptyset,\{(\ell,\ell^t)\},\emptyset)
\end{array}
$$

Fig. 6. Representation Function for the Control Flow Analysis

$(res)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} (\nu n)P$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P$

$(zero)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} \mathbf{0}$ always

$(par)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \mid Q$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \ \wedge\ (\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} Q$

$(repl)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} !P$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P$

$(amb)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} n^{\ell^a}[\,P\,]$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \wedge$
$\forall \ell^a, \ell^{a'} \in \mathbf{Lab}^a : (\ell^a, \ell^{a'}) \in \hat{I} \ \wedge\ (\ell^{a'}, m) \in \hat{H} \ \wedge\ n \in \hat{C}$
$\implies m \in \hat{C}$

$(in)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} \mathbf{in}^{\ell^t} n.P$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \wedge$
$\forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a :$
1: $((\ell^t, \hat{I}) \in \ \wedge\ (\ell^{a''}, \ell^a) \in \hat{I} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I} \ \wedge\ (\ell^{a'}, n) \in \hat{H})$
$\implies (\ell^{a'}, \ell^a) \in \hat{I}$
2: $((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^a, m) \in \hat{H} \ \wedge\ n \in \hat{C})$
$\implies m \in \hat{C}$

$(out)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} \mathbf{out}^{\ell^t} n.P$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \wedge$
$\forall \ell^a, \ell^{a'}, \ell^{a''} \in \mathbf{Lab}^a :$
1: $((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^{a'}, \ell^a) \in \hat{I} \ \wedge\ (\ell^{a''}, \ell^{a'}) \in \hat{I} \ \wedge\ (\ell^{a'}, n) \in \hat{H} \ \wedge$
$(\ell^{a'} \notin \mathbf{Lab}^a_B \vee \ell^a \in \mathbf{Lab}^a_B)) \implies (\ell^{a'}, \ell^a) \in \hat{I}$
2: $((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^a, m) \in \hat{H} \ \wedge\ n \in \hat{C})$
$\implies m \in \hat{C}$

$(open)$ $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} \mathbf{open}^{\ell^t} n.P$ iff $(\hat{C},\hat{I},\hat{H}) \models^{\mathrm{B}} P \wedge$
$\forall \ell^a, \ell^{a'} \in \mathbf{Lab}^a :$
1: $((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^{a'}, n) \in \hat{H} \ \wedge\ (\ell^{a'} \notin \mathbf{Lab}^a_B \vee \ell^a \in \mathbf{Lab}^a_B))$
$\implies \left\{(\ell^a, \ell') \mid (\ell^{a'}, \ell') \in \hat{I}\right\} \subseteq \hat{I}$
2: $((\ell^a, \ell^t) \in \hat{I} \ \wedge\ (\ell^a, m) \in \hat{H} \ \wedge\ n \in \hat{C}) \implies m \in \hat{C}$

Fig. 7. Specification of the Control Flow Analysis