Information Flow Security in Dynamic Contexts*

Riccardo Focardi and Sabina Rossi
Dipartimento di Informatica
Università Ca' Foscari di Venezia
via Torino 155, 30172 Venezia, Italy
{focardi,srossi}@dsi.unive.it

Abstract

We study a security property for processes in dynamic contexts, i.e., contexts that can be reconfigured at run-time. The security property that we propose in this paper, named Persistent_BNDC*, is such that a process is "secure" when every state reachable from it satisfies a basic Non-Interference property. We define a suitable bisimulation based equivalence relation among processes, that allows us to express the new property as a single equivalence check, thus avoiding the universal quantifications over all the reachable states (required by* Persistent_BNDC*) and over all the possible hostile environments (implicit in the basic Non-Interference property we adopt). We show that the novel security property is compositional and we discuss how it can be efficiently checked.*

1. Introduction

In the recent years, systems are becoming more and more complex, and the security community has to face this by considering, e.g., issues like process mobility among different architectures and systems. A mobile process moving on the network can be influenced by the environments it crosses, possibly leading to new security breaches. As an example, consider a mobile agent that collects confidential data (e.g., marketing information) from different commercial hosts. It could be the case that one of the commercial hosts is malicious and tries to deduce some confidential information about the commercial hosts previously visited by the process. We should thus guarantee that the process is protected from the different visited hosts, that could be running different operating systems on different architectures.

In a complex setting like the one described above, we can abstractly think that the (possibly) hostile environment in which a system or an application is running can be dynamically reconfigured at run-time, changing in unpredictable ways. A program executing in a "secure way" inside one environment could find itself in a different setting (with different malicious attackers) at runtime, e.g., if the process decides to migrate during its execution like in the "collecting agent" example above. This could lead to unexpected dangerous situations as a program which is secure for a certain environment might find itself with no protection if the environment itself suddenly changes during its execution.

A number of formal definitions of security properties (see, for instance, [1, 3, 6, 9, 10, 18, 21, 22, 23]) has been proposed in the literature.

In this paper we face the problem of defining a new security property based on the idea of Non-Interference [11, 14, 19, 20, 23] (formalized as *BNDC* [8]), which is suitable to analyze processes in completely dynamic hostile environments. The basic idea is to require that every state which is reachable by the system still satisfies a basic Non-Interference property. If this holds, we are assured that even if the environment changes during the execution no malicious attacker will be able to compromise the system, as every possible reachable state is guaranteed to be secure. This extension of *BNDC*, called *Persistent_BNDC* (*P_BNDC*, for short), leads also to some interesting results, as we are able to prove that it may be equivalently defined by considering the *BNDC* property with a different underlying equivalence notion between processes, i.e., adopting a different discriminating power on processes. This result places this new property in the already studied taxonomy of Non-Interference properties [8] and provides us with a quite efficient way of verifying *P_BNDC*, as it allows us to avoid both the universal quantification over all the possible attackers, which is present in the *BNDC* basic definition, and the universal quantification over all possible reachable states, required by the definition of *P_BNDC* itself. Finally, as we show that *P_BNDC* is equivalent to an already proposed property called *SBSNNI* [8], this work also contributes in giving new verification techniques for

*This work has been partially supported by the MURST projects "Interpretazione astratta, type systems e analisi control-flow" and "Modelli formali per la sicurezza" and the EU project MyThS (IST-2001-32617).

such a property. Compositional properties of *P_BNDC* with respect to the parallel operator and prefix operator are also proved.

The paper is organized as follows. In Section 2, we define the *Security Process Algebra* (SPA) language and recall the notion of *weak bisimulation* over SPA terms. In Section 3, we first give the definitions of *BNDC* and *P_BNDC* and we show that *P_BNDC* is suitable to deal with processes in dynamic contexts. Then we characterize *P_BNDC* through a new definition of *weak bisimulation up to high level actions* and we prove some properties. In Section 4, we report an example of a system which satisfies *P_BNDC* and, finally, in Section 5, we briefly discuss how *P_BNDC* can be efficiently verified and draw some conclusions.

2. The SPA language

The Language. The *Security Process Algebra* (SPA) [8] is a slight extension of Milner's CCS [15], where the set of visible actions is partitioned into high level actions and low level ones in order to specify multilevel systems. SPA syntax is based on the same elements as CCS that is: a set $\mathcal{L}$ of *visible* actions such that $\mathcal{L} = I \cup O$ where $I = \{a, b, \ldots\}$ is a set of *input* actions and $O = \{\bar{a}, \bar{b}, \ldots\}$ is a set of *output* actions; a complementation function $\bar{\cdot} : \mathcal{L} \rightarrow \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$; a special action τ which models internal computations, i.e., not visible outside the system. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. Function $\bar{\cdot}$ is extended to Act by defining $\bar{\tau} = \tau$. In order to obtain a partition of the visible actions into two levels we consider two sets, Act_H and Act_L, of high and low level actions which are closed with respect to $\bar{\cdot}$, i.e., $\overline{Act_H} = Act_H$ and $\overline{Act_L} = Act_L$; moreover they are disjoint and form a covering of $\mathcal{L}$, i.e., $Act_H \cap Act_L = \emptyset$ and $Act_H \cup Act_L = \mathcal{L}$.

The syntax of SPA *agents* (or *processes*) is defined as follows:

$$E ::= \mathbf{0} \mid a.E \mid E + E \mid E|E \mid E \setminus v \mid E[f] \mid Z$$

where $a \in Act$, $v \subseteq \mathcal{L}$, $f : Act \rightarrow Act$ is such that $f(\bar{\alpha}) = \overline{f(\alpha)}$ and $f(\tau) = \tau$, and Z is a constant that must be associated with a definition $Z \stackrel{\text{def}}{=} E$.

Intuitively, $\mathbf{0}$ is the empty process that does nothing; $a.E$ is a process that can perform an action a and then behaves as E; $E_1 + E_2$ represents the non deterministic choice between the two processes E_1 and E_2; $E_1|E_2$ is the parallel composition of E_1 and E_2, where the executions of the two processes are interleaved, possibly synchronized on complementary input/output actions, producing an internal action τ; $E \setminus v$ is a process E prevented from performing actions in v[1]; $E[f]$ is the process E whose actions are renamed *via* the relabelling function f.

[1]Notice that in CCS the operator $\setminus$ requires that the actions of $E \setminus v$ do not belong to $v \cup \bar{v}$.

For the definition of security properties it is also useful the *hiding* operator, $/$, of CSP which can be defined as a relabelling as follows: for a given set $v \subseteq \mathcal{L}$, $E/v \stackrel{\text{def}}{=} E[f_v]$ where $f_v(x) = x$ if $x \notin v$ and $f_v(x) = \tau$ if $x \in v$. In practice, E/v turns all actions in v into internal τ's.

Operational Semantics. Let $\mathcal{E}$ be the set of SPA agents, ranged over by E and F. Let $\mathcal{L}(E)$ denote the *sort* of E, i.e., the set of the (possibly executable) actions occurring syntactically in E. The sets of high level agents and low level ones are defined as $\mathcal{E}_H \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_H \cup \{\tau\}\}$ and $\mathcal{E}_L \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid \mathcal{L}(E) \subseteq Act_L \cup \{\tau\}\}$, respectively. Note that $\mathcal{E}_H \cup \mathcal{E}_L \subset \mathcal{E}$, i.e., there exist systems that execute both high and low level actions allowing communications between the two levels.

The operational semantics of SPA agents is given in terms of *Labelled Transition Systems*. A *Labelled Transition System* (LTS) is a triple $(S, A, \rightarrow)$ where S is a set of states, A is a set of labels (actions), $\rightarrow \subseteq S \times A \times S$ is a set of labelled transitions. The notation $(S_1, a, S_2) \in \rightarrow$ (or equivalently $S_1 \stackrel{a}{\rightarrow} S_2$) means that the system can move from the state S_1 to the state S_2 through the action a. A LTS is *finite* if it has a finite number of states and transitions. The operational semantics of SPA is the LTS $(\mathcal{E}, Act, \rightarrow)$, where the states are the terms of the algebra and the transition relation $\rightarrow \subseteq \mathcal{E} \times Act \times \mathcal{E}$ is defined by structural induction as the least relation generated by the axioms and inference rules reported in Figure 1. The operational semantics for an agent E is the subpart of the SPA LTS reachable from the initial state E.

Observational Equivalence. The concept of *observation equivalence* between two processes is based on the idea that two systems have the same semantics if and only if they cannot be distinguished by an external observer. This is obtained by defining an equivalence relation over states/terms of the SPA LTS, equating two processes when they are indistinguishable. In this way the semantics of a term becomes an equivalence class of terms. In the literature there are various equivalences of this kind. In this paper we consider the *weak bisimulation* equivalence, an observation equivalence which takes care of the nondeterministic structure of the LTSs and focus only on the observable actions.

The general notion of *bisimulation* [15] consists of a mutual step-by-step simulation, i.e., given two processes E and F, when E executes a certain action moving to E' then F must be able to simulate this single step by executing the same action and moving to an agent F' which is again bisimilar to E', and vice-versa. A weak bisimulation is a bisimulation which does not care about internal τ actions, i.e., when F simulates an action of E, it can also execute some τ actions before or after that action.

Prefix	$\dfrac{-}{a.E \xrightarrow{a} E}$
Sum	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'}$
Parallel	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 \mid E_2 \xrightarrow{a} E_1' \mid E_2} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 \mid E_2 \xrightarrow{a} E_1 \mid E_2'} \quad \dfrac{E_1 \xrightarrow{a} E_1' \;\; E_2 \xrightarrow{\bar{a}} E_2'}{E_1 \mid E_2 \xrightarrow{\tau} E_1' \mid E_2'}$
Restriction	$\dfrac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v}$ if $a \notin v$
Relabelling	$\dfrac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$
Constant	$\dfrac{E \xrightarrow{a} E'}{A \xrightarrow{a} E'}$ if $A \stackrel{\text{def}}{=} E$

Figure 1. The operational rules for SPA

We use the following notations. If $t = a_1 \cdots a_n \in Act^*$, then we write $E \xrightarrow{t} E'$ if $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$. We say that E' is reachable from E when there exists $t \in Act^*$ such that $E \xrightarrow{t} E'$. If $a \in Act$, then we write $E \stackrel{a}{\Longrightarrow} E'$ for $E(\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^* E'$ where $(\xrightarrow{\tau})^*$ denotes a (possibly empty) sequence of τ labelled transitions. We also write $E \stackrel{\hat{a}}{\Longrightarrow} E'$ for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow} \equiv (\xrightarrow{\tau})^*$ means zero or more τ labelled transitions).

The notion of *weak bisimulation* is defined as follows.

Definition 2.1 (**Weak Bisimulation**) A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *weak bisimulation* if $(E, F) \in \mathcal{S}$ implies, for all $a \in Act$,

- whenever $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{S}$;
- whenever $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{S}$.

Two agents $E, F \in \mathcal{E}$ are *observation equivalent*, denoted by $E \approx F$, if there exists a weak bisimulation $\mathcal{S}$ containing the pair (E, F).

In [15] it is proved that $\approx$ is the largest weak bisimulation and it is an equivalence relation.

3. Security Properties

In this section, we give some definitions that try to capture every possible information flow from a *classified* (*high*) level of confidentiality to an *untrusted* (*low*) one. A strong requirement of these definitions is that no information flow should be possible even in the presence of malicious processes that run at the classified level. The main motivation is to protect a system also from internal attacks, which could be performed by the so called *Trojan Horse* programs, i.e., programs that are apparently honest but hide inside some malicious code. This programs might be for example downloaded from the network or sent by e-mail, and executed by a high level user at the classified level. In the presence of mobility this becomes of course more and more crucial, as a Trojan Horse program could just enter the system through some transparent mechanism (like, e.g., the download of an applet), and the high level user could never be aware of being executing some downloaded (potentially malicious) code.

The definitions we are going to present are all based on the basic idea of Non-Interference [11]: "No information flow is possible from high to low if what is done at the high

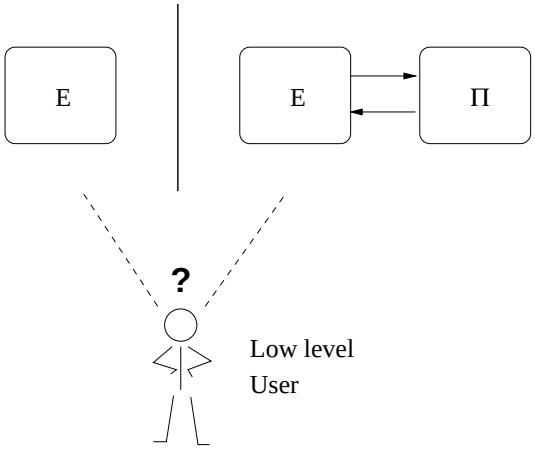

Figure 2. The BNDC property

level *cannot interfere* in any way with the low level".

We start by reporting from [8] the definition of *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short). Then, we extend it in order to deal with dynamic contexts, i.e., by considering the possibility for the hostile environment to change in an unpredictable way. To make the system secure in this setting, we require that every single state of the system is secure by itself. Thus, changing the environment in the middle of a computation will never lead to security breaches.

3.1. Non Deducibility on Compositions

The security property called *Bisimulation-based Non Deducibility on Compositions* (*BNDC*, for short) is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program. A system E is *BNDC* if for every high level process Π a low level user cannot distinguish E from $E|\Pi$. In other words, a system E is *BNDC* if what a low level user sees of the system is not modified by composing any high level process Π to E.

The formal definition of *BNDC* is as follows.

Definition 3.1 (**BNDC**) Let $E \in \mathcal{E}$.

$$E \in BNDC \text{ iff } \forall\, \Pi \in \mathcal{E}_H,\ E \setminus Act_H \approx (E|\Pi) \setminus Act_H.$$

The idea of BNDC is depicted if Figure 2. Let us also show how *BNDC* works through some simple examples.

Example 3.2 First, consider process $E_1 = l_1.h.\bar{l_2}.\mathbf{0}$, where h is the only high level action. It basically accepts the low level input l_1 and then gives $\bar{l_2}$ as output only if h is executed. Note that we have a direct causality between the high level input h and the low level output $\bar{l_2}$, representing a direct information flow. As expected, this system is not *BNDC*. It is sufficient to consider $\Pi = \bar{h}.\mathbf{0}$. In this case it is straightforward to see that $(E_1|\Pi) \setminus Act_H \approx l_1.\bar{l_2}.\mathbf{0}$ while $E_1 \setminus Act_H \approx l_1.\mathbf{0}$. Note that the latter can never execute $\bar{l_2}$, thus $E_1 \setminus Act_H \not\approx (E_1|\Pi) \setminus Act_H$.

The next example aims at showing that *BNDC* is powerful enough to detect information flows due to the possibility for a high level malicious process to block or unblock a system.

Example 3.3 Let us just extend system E_1 as follows: $E_2 = l_1.h.\bar{l_2}.\mathbf{0} + l_1.\bar{l_2}.\mathbf{0}$. We have basically added the trace $l_1.\bar{l_2}$ in order to break the direct causality between h and $\bar{l_2}$. As a matter of fact, now $\bar{l_2}$ may be executed even without h has been previously performed. However, consider the same process $\Pi = \bar{h}.\mathbf{0}$ as before. We again have that $(E_2|\Pi) \setminus Act_H \approx l_1.\bar{l_2}.\mathbf{0}$, but now $E_2 \setminus Act_H \approx l_1.\mathbf{0} + l_1.\bar{l_2}.\mathbf{0} \not\approx (E_2|\Pi) \setminus Act_H$. Note that $E_2 \setminus Act_H$ may (nondeterministically) block after the l_1 input while $(E_2|\Pi) \setminus Act_H$ always executes $\bar{l_2}$. Thus, having many instances of this process, a low level user could deduce if $\bar{h}$ is executed by observing whether the system always performs $\bar{l_2}$ or not. More discussion about how this can be exploited to actually implement a communication channel from high to low may be found in [8]. Process E_2 may be "repaired" and made *BNDC*, by including the possibility of choosing to execute $\bar{l_2}$ or not inside the process, thus completely masking high level activity. Indeed, process $E_3 = l_1.h.\bar{l_2}.\mathbf{0} + l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$ can be proved to be *BNDC*.

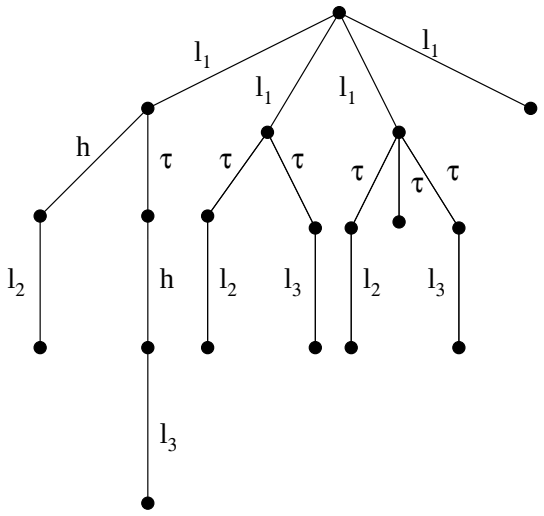

Figure 3. The process $\mathbf{E}_4$

The next example shows that *BNDC* is too weak to detect information flows from *dynamic* (*reconfigurable*) high level malicious processes to low level ones.

Example 3.4 Consider the process $E_4 = l_1.(h.l_2.\mathbf{0} + \tau.h.l_3.\mathbf{0}) + l_1.(\tau.l_2.\mathbf{0} + \tau.l_3.\mathbf{0}) + l_1.(\tau.l_2.\mathbf{0} + \tau.l_3.\mathbf{0} + \tau.\mathbf{0}) + l_1.\mathbf{0}$ depicted in Figure 3, where h is the only high level action. It is easy to prove that E_4 is *BNDC*. Consider also a dynamic high level malicious process that initially behaves as $\bar{h}.\mathbf{0}$ but can be dynamically reconfigured to $\mathbf{0}$ at any computation step. Assume that E_4 is executed in parallel with such a malicious process and that the reconfiguration above happens exactly after two actions of E_4 have been executed. For a low level user, such an execution behaves as the process $l_1.(\tau.l_2.\mathbf{0} + \tau.\mathbf{0}) + l_1.(\tau.l_2.\mathbf{0} + \tau.l_3.\mathbf{0}) + l_1.(\tau.l_2.\mathbf{0} + \tau.l_3.\mathbf{0} + \tau.\mathbf{0}) + l_1.\mathbf{0}$. Note that, after executing l_1, this process may (nondeterministically) move to $(\tau.l_2.\mathbf{0} + \tau.\mathbf{0})$ where it can either execute l_2 or deadlock. The latter is due to the fact that when E_4 is going to perform the second occurrence of h then the high level process is reconfigured to $\mathbf{0}$ blocking the whole execution of the system. In this case a low level user could deduce whether the second occurrence of h in E_4 is performed or not just observing if the system always performs l_3 or not. Hence, in presence of dynamic contexts, E_4 should not be considered "secure".

3.2. Persistent_BNDC

We define a security property which is stronger than *BNDC* but which allows us to deal with possibly dynamic attackers, i.e., high level processes which can be dynamically reconfigured. The novel security property is named *Persistent_BNDC* (*P_BNDC*, for short). The idea is that a system E is *P_BNDC* if for every high level process Π and for every state E' reachable from E a low level user cannot distinguish E' from $E'|\Pi$ (see Figure 4). This is equivalent to say that E is *P_BNDC* if for every state E' reachable from E, E' is *BNDC*.

Formally *P_BNDC* is defined as follows.

Definition 3.5 (**Persistent_BNDC**) Let $E \in \mathcal{E}$.

$$E \in P\_BNDC \text{ iff}$$
$$\forall\ E' \text{ reachable from } E \text{ and } \forall\ \Pi \in \mathcal{E}_H,$$
$$E' \setminus Act_H \approx (E'|\Pi) \setminus Act_H., \text{ i.e., } E' \in BNDC.$$

We show the idea of *P_BNDC* through a simple example.

Example 3.6 Consider again the *BNDC* process E_3 of Example 3.3, $E_3 = l_1.h.\bar{l_2}.\mathbf{0} + l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$. Suppose that it reaches the state $h.\bar{l_2}.\mathbf{0}$ (after executing the first l_1). Now it is clear that this state is not secure, as a direct causality between h and $\bar{l_2}$ is present. In particular $h.\bar{l_2}.\mathbf{0}$ is not *BNDC* and this gives evidence that E_3 is not *P_BNDC*. The process may be "repaired" as follows: $E_5 = l_1.(h.\bar{l_2}.\mathbf{0} + \tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0}) + l_1.(\tau.\bar{l_2}.\mathbf{0} + \tau.\mathbf{0})$. It may be proved that E_5 is *P_BNDC*. Note that, from this example it follows that *P_BNDC* is strictly stronger that *BNDC*, i.e., *P_BNDC* $\subset$ *BNDC*.

3.3. P_BNDC and dynamic contexts

In order to show that the notion of *P_BNDC* is suitable to deal with processes in dynamic contexts we introduce a

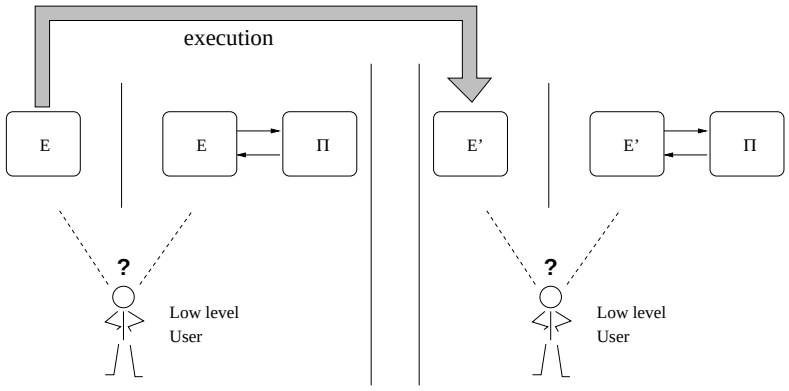

Figure 4. The P_BNDC property

novel bisimulation based equivalence relation, named $\approx_{hc}$, and show that $E \in P\_BNDC$ if and only if E and $E \setminus Act_H$ are not distinguishable with respect to $\approx_{hc}$ for all hostile contexts, i.e., contexts of the form $(\_ \mid \Pi) \setminus Act_H$ where $\Pi \in \mathcal{E}_H$. We show that this is equivalent to say that E running in any *dynamic* hostile context is not distinguishable, for a low level user, from E itself.

Let us first formally introduce the notion of hostile context.

Definition 3.7 (Hostile context) A *hostile context* $C[\_]$ denotes a term of the form $(\_ \mid \Pi) \setminus Act_H$ where $\Pi \in \mathcal{E}_H$, which can be regarded as a mapping from $\mathcal{E}$ to $\mathcal{E}$ that associates with each process $E \in \mathcal{E}$ the process $C[E] \equiv (E|\Pi) \setminus Act_H$.

Observe that for any hostile context $C[\_]$ and process E, all the processes reachable from $C[E]$ have the form $C'[E']$ with $C'[\_]$ being a hostile context too.

We now introduce the concept of *weak bisimulation on hostile contexts*: the idea is that, given two processes E and F, when a hostile context $C[\_]$ filled with E executes a certain action moving E to E' then the same context filled with F is able to simulate this step moving F to F' so that E' and F' are again weakly bisimilar on hostile contexts, and vice-versa. This must be true for every possible hostile context $C[\_]$. It is important to note that the quantification over all possible hostile contexts is re-iterated for E' and F'. This makes this equivalence suitable for dynamic settings in which the environment may change in the middle of system execution.

We use these notations. If $t \in Act^*$ and $C[\_]$ is a hostile context, then we write $E \stackrel{t}{\rightarrow}_C E'$ if $C[E] \stackrel{t}{\rightarrow} C'[E']$; we write $E \stackrel{t}{\Longrightarrow}_C E'$ if $C[E] \stackrel{t}{\Longrightarrow} C'[E']$. Thus, $E \stackrel{\hat{a}}{\Longrightarrow}_C E'$ stands for $C[E] \stackrel{a}{\Longrightarrow} C'[E']$ if $a \in \mathcal{L}$, while it stands for $C[E](\stackrel{\tau}{\rightarrow})^* C'[E']$ if $a = \tau$.

The notion of weak bisimulation on hostile contexts is defined as follows.

Definition 3.8 (Weak Bisimulation on hostile contexts) A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *weak bisimulation on hostile contexts* if $(E, F) \in \mathcal{S}$ implies, for all contexts $C[\_]$ and for all $a \in Act$,

- whenever $E \stackrel{a}{\rightarrow}_C E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_C F'$ and $(E', F') \in \mathcal{S}$;
- whenever $F \stackrel{a}{\rightarrow}_C F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow}_C E'$ and $(E', F') \in \mathcal{S}$.

We say that two processes $E, F \in \mathcal{E}$ are *weakly bisimilar on hostile contexts*, written $E \approx_{hc} F$, if $(E, F) \in \mathcal{S}$ for some weak bisimulation on hostile contexts $\mathcal{S}$. This may be equivalently expressed as follows:

$$\approx_{hc} \;=\; \bigcup\{\mathcal{S} : \; \mathcal{S} \text{ is a weak bisimulation on hostile contexts}\}.$$

It is easy to prove that

- $\approx_{hc}$ is the largest weak bisimulation on hostile contexts
- $\approx_{hc}$ is an equivalence relation.

The next Theorem gives a characterization of *P_BNDC* processes in terms of $\approx_{hc}$.

Theorem 3.9 Let $E \in \mathcal{E}$.
Then, $E \in P\_BNDC$ iff $E \setminus Act_H \approx_{hc} E$.

PROOF. See Appendix. □

The next result allows us to state that the notion of *P_BNDC* is suitable to deal with processes in dynamic contexts.

Let us first give a definition of *dynamic hostile context*, i.e., a hostile context where the high level component may arbitrarily change at any computation step.

Definition 3.10 (**Dynamic hostile context**) A *dynamic hostile context* $C_{dyn}[\_]$ denotes a term $(\_ \mid \Pi)_{dyn} \setminus Act_H$ with $\Pi \in \mathcal{E}_H$ such that for all $a \in Act$, whenever $(E|\Pi) \setminus Act_H \xrightarrow{a} (E'|\Pi') \setminus Act_H$ then $C_{dyn}[E] \xrightarrow{a} C''_{dyn}[E']$ where $C''_{dyn}[\_] = (\_ \mid \Pi'')_{dyn} \setminus Act_H$ for some $\Pi'' \in \mathcal{E}_H$.

Theorem 3.11 Let $E \in \mathcal{E}$.
If $E \setminus Act_H \approx_{hc} E$ then $C_{dyn}[E] \approx E \setminus Act_H$, for all dynamic hostile contexts $C_{dyn}[\_]$.

PROOF. See Appendix. □

Corollary 3.12 Let $E \in \mathcal{E}$.
If $E \in P\_BNDC$ then $C_{dyn}[E] \approx E \setminus Act_H$, for all dynamic hostile contexts $C_{dyn}[\_]$.

3.4. Avoiding the universal quantifications

We show now how it is possible to give a characterization of *P_BNDC* avoiding both the universal quantification over all the possible high level processes, which is present in the *BNDC* basic definition, and the universal quantification over all the possible reachable states, required by the definition of *P_BNDC* itself.

In the previous subsection, we have shown how the idea of "being secure in every state" can be directly moved inside the bisimulation equivalence notion. However, the notion of weak bisimulation on hostile contexts implicitly contains a quantification over all possible malicious contexts. We show here that the same equivalence notion, i.e., $\approx_{hc}$, may be expressed in a rather simpler way by exploiting local information only. This can be done by defining a novel equivalence relation which focusses only on observable actions that do not belong to Act_H.

More in detail, we define an observation equivalence where actions from Act_H *may* be ignored, i.e., they may be matched by zero or more τ actions. To this aim, we use a transition relation which does not take care of both internal actions and actions from Act_H as follows.

Definition 3.13 For an action $a \in Act$, we write $(\xrightarrow{a})^{\{0,1\}}$ to denote a sequence of zero or one a actions. The expression $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ is a shorthand for $E \stackrel{\hat{a}}{\Longrightarrow} E'$ if $a \notin Act_H$, and for $E(\xrightarrow{\tau})^*(\xrightarrow{a})^{\{0,1\}}(\xrightarrow{\tau})^*E'$ if $a \in Act_H$.

Notice that the relation $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H}$ is a generalization of the relation $\stackrel{\hat{a}}{\Longrightarrow}$ used in the definition of weak bisimulation [15]. In fact, if $Act_H = \emptyset$ then for all $a \in Act$, $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ coincides with $E \stackrel{\hat{a}}{\Longrightarrow} E'$.

We define the concept of weak bisimulation up to Act_H.

Definition 3.14 (**Weak Bisimulation up to** Act_H) A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over agents is a *weak bisimulation up to* Act_H if $(E, F) \in \mathcal{S}$ implies, for all $a \in Act$,

- whenever $E \xrightarrow{a} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $(E', F') \in \mathcal{S}$;
- whenever $F \xrightarrow{a} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} E'$ and $(E', F') \in \mathcal{S}$.

We say that two agents $E, F \in \mathcal{E}$ are *weakly bisimilar up to* Act_H, written $E \approx_{\setminus Act_H} F$, if $(E, F) \in \mathcal{S}$ for some weak bisimulation $\mathcal{S}$ up to Act_H. This is equivalently expressed as follows:

$$\approx_{\setminus Act_H} = \bigcup \{\mathcal{S} : \mathcal{S} \text{ is a weak bisimulation up to } Act_H\}.$$

It is easy to prove that

- $\approx_{\setminus Act_H}$ is the largest weak bisimulation up to Act_H
- $\approx_{\setminus Act_H}$ is an equivalence relation.

The next theorem shows that the relations $\approx_{hc}$ and $\approx_{\setminus Act_H}$ are equivalent.

Theorem 3.15 Let $E, F \in \mathcal{E}$.
Then, $E \approx_{hc} F$ iff $E \approx_{\setminus Act_H} F$.

PROOF. See Appendix. □

Theorem 3.15 allows us to identify a local property of processes (with no quantification on the states and on the hostile contexts) which is a necessary and sufficient condition for *P_BNDC*. This is stated by the following corollary.

Corollary 3.16 Let $E \in \mathcal{E}$.
$E \in P\_BNDC$ iff $E \setminus Act_H \approx_{\setminus Act_H} E$.

In practice, we have proven that a process is *P_BNDC* if and only if it is equivalent – with respect to a particular bisimulation based equivalence relation – to the same process prevented from performing high level actions. This property is particularly appealing since it suggests the effective computability of *P_BNDC*. In particular, as we discuss in the concluding section, one may perform the *P_BNDC* check using already existing tools at a low time complexity.

3.5. Properties of P_BNDC

In this subsection we show that property *P_BNDC* is equivalent to the already proposed security property *SBSNNI* (Strong Bisimulation-based SNNI, where *SNNI* stands for Strong Non-deterministic Non Interference, see [7, 8]) and we prove that it is compositional with respect to both parallel and prefix operators.

The security property *SBSNNI* was defined in [7, 8] as follows.

Definition 3.17 (**SBSNNI**) Let $E \in \mathcal{E}$.

$$E \in SBSNNI \text{ iff}$$
$$\forall\, E' \text{ reachable from } E,\ E' \setminus Act_H \approx E'/Act_H.$$

This property was introduced to automatically check *BNDC*, i.e., to bypass the quantification over all the possible malicious high level processes. As it follows from the next proposition, *SBSNNI* is strictly stronger than *BNDC*, since, quite interestingly, it is equivalent to *P_BNDC*.

Proposition 3.18 $P\_BNDC = SBSNNI$.

PROOF. See Appendix. □

In [8] it is proved that *SBSNNI* is *compositional*, in the sense that it is preserved by the parallel and restriction operators (statements (1) and (2) of Proposition 3.19). It is easy to prove that P_BNDC is also compositional with respect to the prefix operator limited to low level actions (statement (3) of Proposition 3.19).

Proposition 3.19

(1) if $E, F \in P\_BNDC$ then $(E|F) \in P\_BNDC$,

(2) if $E \in P\_BNDC$ and $v \subseteq \mathcal{L}$ then $E \setminus v \in P\_BNDC$,

(3) if $E \in P\_BNDC$ and $a \in Act_L \cup \{\tau\}$ then $a.E \in P\_BNDC$.

4. An example

In this section we report from [7, 8] a non trivial example of a system which is *SBSNNI* and thus *P_BNDC*. Our aim is to give evidence that the proposed property is not too restrictive. Moreover, in [7, 8], *SBSNNI* was used to prove that the system was *BNDC* (as a sufficient condition). Following the intuition of *P_BNDC* we can now state that such a system is secure even when its execution environment is dynamic and changes at runtime. The system itself could be seen as a very simple mobile "collecting agent" (as the one described in the introduction) that may be accessed in different hosts.

More precisely, the example reported here is an access monitor which handles read and write requests on two binary variables enforcing the Multilevel Security Policy [2], a particular access control policy which has the aim of ensuring that no information flow is possible from high level to low one. The policy is based on two access rules: *no read up*, i.e., no subject can read from an object with a higher level; *no write down*, i.e., no subject can write to an object with a lower level. In particular, the access monitor process handles read and write requests from high and low level users on two binary objects: a high level variable and a low level one. It achieves *no read up* and *no write down* access control rules allowing a high level user to read from both objects and write only on the high one; conversely, a low level user is allowed to write on both objects and read only from the low one.

The access monitor system [2] is reported in Figure 5 (see also Figure 6). In such a system we have that $k \in \{0, 1, \mathit{err}\}$, $L = \{r, w\}$, $N = \{\mathit{val}, \mathit{access\_r}, \mathit{access\_w}\}$ and $a\_r(1, x)$, $a\_w(1, x)$, $\mathit{put}(1, y) \in Act_H\ \forall x \in \{0, 1\}$ and $\forall y \in \{0, 1, \mathit{err}\}$, while the same actions with 0 as first parameter belong to Act_L.

Users interact with the monitor through the following access actions:

- $a\_r(l, x)$, a read request from level l to object x;
- $a\_w(l, x, z)$, a write request from level l to object x of value z;
- $\mathit{put}(l, y)$, the response to level l for a previous read request; y is the returned (read) value.

where l is the user level ($l = 0$ low, $l = 1$ high), x is the object ($x = 0$ low, $x = 1$ high) and z is the binary value to be written.

As an example, consider $a\_r(0, 1)$ which represents a low level user ($l = 0$) read request from the high level object ($x = 1$), and $a\_w(1, 0, 0)$ which represents a high level user ($l = 1$) write request of value 0 ($z = 0$) on the low object ($x = 0$). Read results are returned to users through the output actions $\mathit{put}(l, y)$. This can be also an error in case of a read-up request. Note that if a high level user tries to write on the low object – through $\mathit{access\_w}(1, 0, z)$ – such a request is not executed and no error message is returned.

The *Access_Monitor* is the parallel composition of the actual monitor *AM* and an interface for each level which temporarily stores the output value of the monitor (passing it later to the users and thus making communication asynchronous) and that guarantees mutual exclusion within the same level. This interface is crucial to guarantee *P_BNDC*

[2] Note that the system is specified using a value-passing extension of SPA. We will briefly explain its translation to the "pure" calculus in the following.

$$
\begin{aligned}
\mathit{Access\_Monitor} &= (\mathit{AM} \mid \mathit{Interf}) \setminus N \\
\mathit{AM} &= (\mathit{Monitor} \mid \mathit{Object}(1,0) \mid \mathit{Object}(0,0)) \setminus L \\
\mathit{Monitor} &= \mathit{access\_r}(l,x). \\
&\quad (\textbf{ if } x \leq l \textbf{ then } r(x,y).\overline{\mathit{val}}(l,y).\mathit{Monitor} \\
&\quad \textbf{ else } \overline{\mathit{val}}(l,\mathit{err}).\mathit{Monitor}) \\
&\quad + \\
&\quad \mathit{access\_w}(l,x,z). \\
&\quad (\textbf{ if } x \geq l \textbf{ then } \overline{w}(x,z).\mathit{Monitor} \\
&\quad \textbf{ else } \mathit{Monitor}) \\
\mathit{Object}(x,y) &= \overline{r}(x,y).\mathit{Object}(x,y) + w(x,z).\mathit{Object}(x,z) \\
\mathit{Interf} &= \mathit{Interf}(0) \mid \mathit{Interf}(1) \\
\mathit{Interf}(l) &= a\_r(l,x).\overline{\mathit{access\_r}}(l,x).\mathit{val}(l,k).\overline{\mathit{put}}(l,k).\mathit{Interf}(l) \\
&\quad + \\
&\quad a\_w(l,x,z).\overline{\mathit{access\_w}}(l,x,z).\mathit{Interf}(l)
\end{aligned}
$$

Figure 5. The Access_Monitor System.

property as without it a high level user could block the monitor process indefinitely, by never accepting the response of a read request (output value) leading to an indirect information flow.

In order to understand how the system works, let us consider the following transitions sequence representing the writing of value 1 in the low level object, performed by the low level user:

$(\mathit{AM} \mid \mathit{Interf}(0) \mid \mathit{Interf}(1)) \setminus N$

$\xrightarrow{a\_w(0,0,1)}$

$(\mathit{AM} \mid \overline{\mathit{access\_w}}(0,0,1).\mathit{Interf}(0) \mid \mathit{Interf}(1)) \setminus N$

$\xrightarrow{\tau}$

$((\overline{w}(0,1).\mathit{Monitor} \mid \mathit{Object}(1,0) \mid \mathit{Object}(0,0)) \setminus L \mid \mathit{Interf}) \setminus N$

$\xrightarrow{\tau}$

$((\mathit{Monitor} \mid \mathit{Object}(1,0) \mid \mathit{Object}(0,1)) \setminus L \mid \mathit{Interf}) \setminus N$

The trace corresponding to this sequence of transitions is

$$a\_w(0,0,1)$$

and so we can write:

$(\mathit{AM} \mid \mathit{Interf}(0) \mid \mathit{Interf}(1)) \setminus N$

$\stackrel{a\_w(0,0,1)}{\Longrightarrow}$

$((\mathit{Monitor} \mid \mathit{Object}(1,0) \mid \mathit{Object}(0,1)) \setminus L \mid \mathit{Interf}) \setminus N$

Note that, after the execution of the trace, the low level object contains value 1.

Access_Monitor is a value passing specification of an access monitor. Its translation into pure SPA is reported and described in detail in [7, 8]. The idea is to translate each possible combination of the values into different SPA actions and different SPA processes. As an example, here we provide the translation of $Object(x,y)$ into the pure calculus by means of the following four constant definitions:

$$
\begin{aligned}
\mathit{Object}_{00} &= \overline{r}_{00}.\mathit{Object}_{00} + w_{00}.\mathit{Object}_{00} + w_{01}.\mathit{Object}_{01} \\
\mathit{Object}_{01} &= \overline{r}_{01}.\mathit{Object}_{01} + w_{00}.\mathit{Object}_{00} + w_{01}.\mathit{Object}_{01} \\
\mathit{Object}_{10} &= \overline{r}_{10}.\mathit{Object}_{10} + w_{10}.\mathit{Object}_{10} + w_{11}.\mathit{Object}_{11} \\
\mathit{Object}_{11} &= \overline{r}_{11}.\mathit{Object}_{11} + w_{10}.\mathit{Object}_{10} + w_{11}.\mathit{Object}_{11}
\end{aligned}
$$

Note that we have, for every possible value of the pair (x,y), one different process Object_{xy} and two different actions $\overline{r}_{xy}$ and w_{xy}.

5. Conclusion

In this paper we studied a security property, named *Persistent_BNDC*, which is based on the idea of Non-Interference and it is suitable to deal with processes in dynamic environments. We characterized *P_BNDC* through a local property (with no quantification on the states and on the hostile contexts) by using a new definition of *weak bisimulation up to high level actions*, denoted by $\approx_{\setminus Act_H}$.

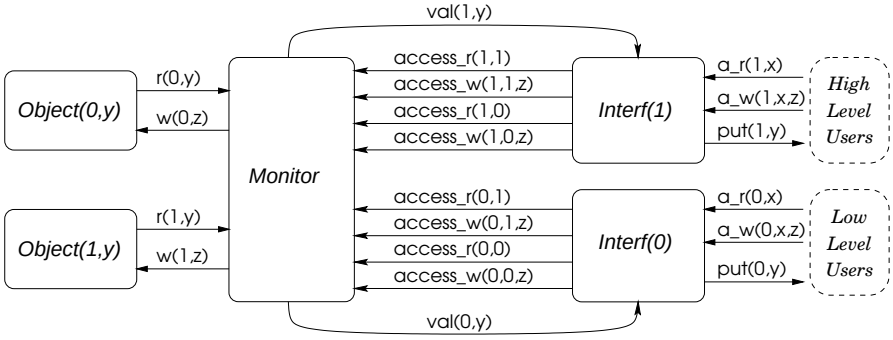

Figure 6. The P_BNDC Access_Monitor

This result is interesting since it allows us to reduce the problem of verifying *P_BNDC* to the problem of checking a weak bisimulation up to high level actions between two processes. In the case of finite state processes, this can be efficiently done either adopting the model-checking technique or using a strong bisimulation checker. The model-checking technique can be used as follows: one can exploit the well-known greatest fixpoint characterization of bisimulation-like relations [16] to derive modal mu-calculus formulae characterizing finite-state processes up to the equivalence relation $\approx_{\setminus Act_H}$. In this case model checkers can be employed as *P_BNDC* checkers. Indeed, if $\phi^{\approx_{\setminus Act_H}}$ is a characteristic formulae for a finite state process E up to $\approx_{\setminus Act_H}$, then $E \in P\_BNDC$ if and only if $E \setminus Act_H \models \phi^{\approx_{\setminus Act_H}}$ (see [24, 25]). *P_BNDC* can be also proved by following the method proposed in [24] where the verification of a process equivalence is reduced to the problem of verifying a strong bisimulation between two transformed processes. Given this transformation, the strong bisimulation test can be performed using efficient algorithms for strong bisimulation ([17, 12, 4, 13, 5]).

Actually, the compositional security checker described in [7] provides an automatic tool for verifying *P_BNDC* over finite state processes: this is done by checking *SBSNNI* that requires to verify a bisimulation property over all the possible reachable states. The results presented in this paper show us how we may improve the *P_BNDC* checker by extending our compositional checker in order to deal with the novel bisimulation-like equivalence relation $\approx_{\setminus Act_H}$. We plan to work on this subject in the next future.

Property *P_BNDC* is not compositional with respect to the nondeterministic choice operator. We are exploring the existence of a (possibly) large class of *P_BNDC* processes for which the *P_BNDC* property is preserved also through such operator.

Finally, we observe that *P_BNDC* is a property suitable for reasoning on security of processes in presence of mobility. However the language upon which we study such a property is not exactly a language for mobility. Thus the next step will be to study *P_BNDC* (or similar properties) when more suitable languages for dealing with mobility are considered.

Acknowledgements.

We would like to thank Michele Bugliesi and Fabio Martinelli for discussions on the *BNDC* property and the concept of security in the presence of dynamic contexts.

References

[1] M. Abadi. Secrecy by Typing in Security Protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] D. E. Bell and L. J. L. Padula. Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306, MITRE MTR-2997, 1976.

[3] C. Bodei, P. Degano, R. Focardi, and C. Priami. Primitives for Authentication in Process Algebras. *Theoretical Computer Science*, 2001. To appear.

[4] A. Bouali and R. de Simone. Symbolic bisimulation minimization. In G. von Bochmann and D. K. Probst, editors, *Proc. of International Conference on Computer Aided Verification (CAV'92)*, volume 663 of *LNCS*, pages 96–108. Springer-Verlag, Berlin, 1992.

[5] A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 79–90. Springer-Verlag, Berlin, 2001.

[6] N. A. Durgin, J. C. Mitchell, and D. Pavlovic. A Compositional Logic for Protocol Correctness. In *Proc. of of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2001.

[7] R. Focardi and R. Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow

Security Properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

[8] R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*. Springer-Verlag, Berlin, 2001.

[9] R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In U. Montanari, J. Rolim, and E. Welzl, editors, *Proc. of International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 744–755. Springer-Verlag, Berlin, 2000.

[10] R. Focardi and F. Martinelli. A Uniform Approach for the Definition of Security Properties. In J. Wing, J. Woodcook, and J. Davies, editors, *Proc. of World Congress on Formal Methods (FM'99)*, volume 1708 of *LNCS*, pages 794–813. Springer-Verlag, Berlin, 1999.

[11] J. A. Goguen and J. Meseguer. Security Policy and Security Models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[12] P. C. Kannellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[13] D. Lee and M. Yannakakis. Online minimization of transition systems. In *Proc. of the 24th ACM Symposium on Theory of Computing (STOC'92)*, pages 264–274. ACM Press, 1992.

[14] G. Lowe. Defining information flow. Technical Report 1999/3, Department of Mathematics and Computer Science, University of Leicester, 1999.

[15] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[16] M. Müller-Olm. Derivation of Characteristic Formulae. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[17] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[18] L. C. Paulson. Proving Properties of Security Protocols by Induction. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.

[19] P. Ryan and S. Schneider. Process algebra and noninterference. *Journal of Computer Security*, 9(1/2):75–103, 2001.

[20] A. Sabelfed and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 200–215. IEEE Computer Society Press, 2000.

[21] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), 1998.

[22] V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. In *Proc. of 7th Annual Symposium on Network and Distributed System Security (NDSS 2000)*, pages 119–128. Internet Society, 2000.

[23] G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL98)*, pages 355–364. ACM Press, 1998.

[24] B. Steffen and A. Ingòlfsdòttir. Characteristic Formulae for Processes with Divergence. *Information and Computation*, 110(1):149–163, 1994.

[25] C. Stirling. Modal and Temporal Logics for Processes. In E. Brinksma, R. Cleaveland, K. G. T. M. Larsen, and B. Steffen, editors, *Logics for Concurrency: Structures versus Automata*, volume 1043 of *LNCS*, pages 149–237. Springer-Verlag, Berlin, 1996.

A. Proofs

In this Appendix we give the proofs of Theorems 3.9, 3.11 and 3.15 and Proposition 3.18.

In the proof of Theorem 3.9 we use the following lemma which easily follows from the definitions of hostile context and of weak bisimulation on hostile contexts.

Lemma A.1 Let $E \in \mathcal{E}$ such that $E \approx_{hc} E \setminus Act_H$. Then for all E' reachable from E there exists $E'' \setminus Act_H$ reachable from $E \setminus Act_H$ such that $E' \approx_{hc} E'' \setminus Act_H$.

Proof of Theorem 3.9. We first show that $E \setminus Act_H \approx_{hc} E$ implies $E \in P\_BNDC$. In order to do it we prove that

$$\begin{array}{rcl} \mathcal{S} & = & \{(E_1 \setminus Act_H, (E_2|\Pi) \setminus Act_H) \mid \\ & & \qquad \Pi \in \mathcal{E}_H \text{ and } E_1 \setminus Act_H \approx_{hc} E_2\} \end{array}$$

is a weak bisimulation.
This is sufficient to say that $E \in P\_BNDC$. In fact, by Lemma A.1, for every state E' reachable from E there exists a state $E'' \setminus Act_H$ reachable from $E \setminus Act_H$ such that $E'' \setminus Act_H \approx_{hc} E'$. Hence, by definition of $\mathcal{S}$, we have that for all $\Pi \in \mathcal{E}_H$, $(E'' \setminus Act_H, (E'|\Pi) \setminus Act_H) \in \mathcal{S}$. Since $\mathcal{S}$ is a weak bisimulation, we have that for all $\Pi \in \mathcal{E}_H$, $E'' \setminus Act_H \approx (E'|\Pi) \setminus Act_H$ and, in particular, $E'' \setminus Act_H \approx E' \setminus Act_H$. Since $\approx$ is an equivalence relation, by symmetry and transitivity, we have that for every E' reachable from E and for all $\Pi \in \mathcal{E}_H$, $E' \setminus Act_H \approx (E'|\Pi) \setminus Act_H$, i.e., $E \in P\_BNDC$.

The fact that $\mathcal{S}$ is a weak bisimulation follows from the following four cases. Let $(E_1 \setminus Act_H, (E_2|\Pi) \setminus Act_H) \in \mathcal{S}$.

Case 1. $E_1 \setminus Act_H \stackrel{a}{\rightarrow} E_1' \setminus Act_H$ with $a \notin Act_H$. Thus, for all contexts $C[\_]$, $E_1 \setminus Act_H \stackrel{a}{\rightarrow}_C E_1' \setminus Act_H$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, for all contexts $C[\_]$ there exists E_2' such that $E_2 \stackrel{\hat{a}}{\Longrightarrow}_C E_2'$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Hence there exists E_2' such that $(E_2|\Pi) \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} (E_2'|\Pi') \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E_1' \setminus Act_H, (E_2'|\Pi') \setminus Act_H) \in \mathcal{S}$.

Case 2. $(E_2|\Pi) \setminus Act_H \stackrel{a}{\rightarrow} (E_2'|\Pi) \setminus Act_H$ where also $E_2 \setminus Act_H \stackrel{a}{\rightarrow} E_2' \setminus Act_H$ and $a \notin Act_H$. Let $C[\_]$ be the context $(\_|\mathbf{0}) \setminus Act_H$. Hence $E_2 \stackrel{a}{\rightarrow}_C E_2'$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow}_C E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E_1' \setminus Act_H$ and $(E_1' \setminus Act_H, (E_2'|\Pi) \setminus Act_H) \in \mathcal{S}$.

Case 3. $(E_2|\Pi) \setminus Act_H \stackrel{\tau}{\rightarrow} (E_2|\Pi') \setminus Act_H$ with $\Pi \stackrel{\tau}{\rightarrow} \Pi'$. By definition of $\mathcal{S}$, it follows trivially that $(E_1 \setminus Act_H, (E_2|\Pi') \setminus Act_H) \in \mathcal{S}$.

Case 4. $(E_2|\Pi) \setminus Act_H \stackrel{\tau}{\rightarrow} (E_2'|\Pi') \setminus Act_H$ where $E_2 \stackrel{a}{\rightarrow} E_2'$, $\Pi \stackrel{\bar{a}}{\rightarrow} \Pi'$ and $a \in Act_H$. Let $C[\_]$ be the context $(\_|\bar{a}) \setminus Act_H$. Hence $E_2 \stackrel{\tau}{\rightarrow}_C E_2'$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow}_C E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow} E_1' \setminus Act_H$ and $(E_1' \setminus Act_H, (E_2'|\Pi) \setminus Act_H) \in \mathcal{S}$.

We now show that if $E \in P\_BNDC$ then $E \setminus Act_H \approx_{hc} E$. To this end it is sufficient to prove that

$$\begin{array}{rcl} \mathcal{S} & = & \{(E_1 \setminus Act_H, E_2) \mid E_1 \setminus Act_H \approx E_2 \setminus Act_H \\ & & \qquad \text{and } E_2 \in P\_BNDC\} \end{array}$$

is a weak bisimulation on hostile contexts. This follows from the following three cases.

Let $C[\_]$ be an hostile context.

Case 1. $E_1 \setminus Act_H \stackrel{a}{\rightarrow}_C E_1' \setminus Act_H$. Then it also holds $E_1 \setminus Act_H \stackrel{a}{\rightarrow} E_1' \setminus Act_H$. From the fact that $E_1 \setminus Act_H \approx E_2 \setminus Act_H$, we have that there exists E_2' such that $E_2 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E_2' \setminus Act_H$ and $E_1' \setminus Act_H \approx E_2' \setminus Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E_2' \in P\_BNDC$. From the fact that $E_2 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E_2' \setminus Act_H$ we have that $E_2 \stackrel{\hat{a}}{\Longrightarrow} E_2'$ and thus $E_2 \stackrel{\hat{a}}{\Longrightarrow}_C E_2'$ and, by definition of $\mathcal{S}$, $(E_1' \setminus Act_H, E_2') \in \mathcal{S}$.

Case 2. $E_2 \stackrel{a}{\rightarrow}_C E_2'$ with $E_2 \setminus Act_H \stackrel{a}{\rightarrow} E_2' \setminus Act_H$. Since $E_1 \setminus Act_H \approx E_2 \setminus Act_H$, there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx E_2' \setminus Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E_2' \in P\_BNDC$. Hence $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow}_C E_1' \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E_1' \setminus Act_H, E_2') \in \mathcal{S}$.

Case 3. $E_2 \stackrel{\tau}{\rightarrow}_C E_2'$ with $E_2 \stackrel{a}{\rightarrow} E_2'$ and $a \in Act_H$. Then, $E_2/Act_H \stackrel{\tau}{\rightarrow} E_2'/Act_H$. From the fact that $E_1 \setminus Act_H \approx E_2 \setminus Act_H$ and $E_2 \in P\_BNDC$, we have that $E_1 \setminus Act_H \approx E_2/Act_H$ and thus there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow} E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx E_2'/Act_H$. Moreover, since $E_2 \in P\_BNDC$ also $E_2' \in P\_BNDC$ and hence $E_1' \setminus Act_H \approx E_2' \setminus Act_H$. Thus $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow}_C E_1' \setminus Act_H$ and, by definition of $\mathcal{S}$, $(E_1' \setminus Act_H, E_2') \in \mathcal{S}$. □

Proof of Theorem 3.11. We show that $E \setminus Act_H \approx_{hc} E$ implies $C_{dyn}[E] \approx E \setminus Act_H$, for all dynamic hostile contexts $C_{dyn}[\_]$. In order to do it we prove that

$$\begin{array}{rcl} \mathcal{S} & = & \{(E_1 \setminus Act_H, C_{dyn}[E_2]) \mid E_1 \setminus Act_H \approx_{hc} E_2 \\ & & \qquad \text{and } C_{dyn}[\_] \text{ is a dynamic context}\} \end{array}$$

is a weak bisimulation.
This is clearly sufficient to say that $C_{dyn}[E] \approx E \setminus Act_H$, for all dynamic hostile contexts $C_{dyn}[\_]$.

The fact that $\mathcal{S}$ is a weak bisimulation follows from the following four cases.

Let $(E_1 \setminus Act_H, C_{dyn}[E_2]) \in \mathcal{S}$.

Case 1. $E_1 \setminus Act_H \stackrel{a}{\rightarrow} E_1' \setminus Act_H$ with $a \notin Act_H$. Thus, for all hostile contexts $C[\_]$, we have $E_1 \setminus Act_H \stackrel{a}{\rightarrow}_C E_1' \setminus Act_H$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, for all hostile contexts $C[\_]$ there exists E_2' such that $E_2 \stackrel{\hat{a}}{\Longrightarrow}_C E_2'$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Hence there exists E_2' such that $C_{dyn}[E_2] \stackrel{\hat{a}}{\Longrightarrow} C'_{dyn}[E_2']$ for some dynamic hostile context $C_{dyn}[\_]$, and then, by definition of $\mathcal{S}$, $(E_1' \setminus Act_H, C'_{dyn}[E_2']) \in \mathcal{S}$.

Case 2. $C_{dyn}[E_2] \stackrel{a}{\rightarrow} C'_{dyn}[E_2']$ with $E_2 \setminus Act_H \stackrel{a}{\rightarrow} E_2' \setminus Act_H$ and $a \notin Act_H$. Let $C[\_] \equiv (\_|\mathbf{0}) \setminus Act_H$. Hence $E_2 \stackrel{a}{\rightarrow}_C E_2'$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow}_C E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have $E_1 \setminus Act_H \stackrel{\hat{a}}{\Longrightarrow} E_1' \setminus Act_H$ and $(E_1' \setminus Act_H, C'_{dyn}[E_2']) \setminus Act_H) \in \mathcal{S}$.

Case 3. $C_{dyn}[E_2] \stackrel{a}{\rightarrow} C'_{dyn}[E_2]$. By definition of $\mathcal{S}$, it follows trivially that $(E_1 \setminus Act_H, C'_{dyn}[E_2]) \in \mathcal{S}$.

Case 4. $C_{dyn}[E_2] \stackrel{a}{\rightarrow} C'_{dyn}[E_2']$ where $E_2 \stackrel{a}{\rightarrow} E_2'$ and $a \in Act_H$. Let $C[\_] \equiv (\_|\bar{a}) \setminus Act_H$. Hence $E_2 \stackrel{\tau}{\rightarrow}_C E_2'$. By the hypothesis that $E_1 \setminus Act_H \approx_{hc} E_2$, there exists E_1' such that $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow}_C E_1' \setminus Act_H$ and $E_1' \setminus Act_H \approx_{hc} E_2'$. Since $C[E_1 \setminus Act_H]$ can only perform actions of $E_1 \setminus Act_H$, we have $E_1 \setminus Act_H \stackrel{\hat{\tau}}{\Longrightarrow} E_1' \setminus Act_H$ and $(E_1' \setminus Act_H, C'_{dyn}[E_2']) \in \mathcal{S}$. □

Proof of Theorem 3.15. We first show that $E \approx_{hc} F$ implies $E \approx_{\setminus Act_H} F$. To this end it is sufficient to prove that

$$\mathcal{S} \quad = \quad \{(E, F) \mid \quad E \approx_{hc} F\}$$

is a weak bisimulation up to Act_H.
This follows from the following two cases.

Case 1. $E \stackrel{a}{\rightarrow} E'$ with $a \notin Act_H$. Let $C[\_]$ be the hostile context $(\_|\mathbf{0}) \setminus Act_H$. Then $E \stackrel{a}{\rightarrow}_C E'$. From the fact that $E \approx_{hc} F$ it follows that there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_C F'$ and $E' \approx_{hc} F'$. By the choice of C, we also have that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and, since $a \notin Act_H$, $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$. Moreover, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

Case 2. $E \stackrel{a}{\rightarrow} E'$ with $a \in Act_H$. Let $C[\_]$ be the hostile context $\equiv (\_|\bar{a}) \setminus Act_H$. Then $E \stackrel{\tau}{\rightarrow}_C E'$. From the fact that $E \approx_{hc} F$ it follows that there exists F' such that $F \stackrel{\hat{\tau}}{\Longrightarrow}_C F'$ and $E' \approx_{hc} F'$. By the choice of C, we also have that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

We now show that $E \approx_{\setminus Act_H} F$ implies $E \approx_{hc} F$. To this end it is sufficient to prove that

$$\mathcal{S} \quad = \quad \{(E, F) \mid \quad E \approx_{\setminus Act_H} F\}$$

is a weak bisimulation on hostile contexts.
This follows from the following two cases.

Let $C[\_]$ be a hostile context.

Case 1. $E \stackrel{a}{\rightarrow}_C E'$ with $E \stackrel{a}{\rightarrow} E'$ and $a \notin Act_H$. Since $E \approx_{\setminus Act_H} F$, there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $E' \approx_{\setminus Act_H} F'$. Since $a \notin Act_H$, we also have $F \stackrel{\hat{a}}{\Longrightarrow} F'$. Thus $F \stackrel{\hat{a}}{\Longrightarrow}_C F'$ and, by definition of $\mathcal{S}$, $(E', F') \in \mathcal{S}$.

Case 2. $E \stackrel{\tau}{\rightarrow}_C E'$ with $E \stackrel{a}{\rightarrow} E'$ and $a \in Act_H$. Since $E \approx_{\setminus Act_H} F$, there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow}_{\setminus Act_H} F'$ and $E' \approx_{\setminus Act_H} F'$. Thus either $F \stackrel{\hat{\tau}}{\Longrightarrow} F'$ or $F \stackrel{\hat{a}}{\Longrightarrow} F'$. Since the hostile context $C[\_]$ may synchronize on a by performing the complementary action $\bar{a}$, we have that $F \stackrel{\hat{\tau}}{\Longrightarrow}_C F'$ and $(E', F') \in \mathcal{S}$. □

Before proving Proposition 3.18 we recall from [8] the next definition and result.

Definition A.2 [8] Let $E \in \mathcal{E}$.
Then $E \in BSNNI$ iff $E \setminus Act_H \approx E/Act_H$.

Proposition A.3 [8] $BNDC \subset BSNNI$.

Proof of Proposition 3.18 We first prove that $P\_BNDC \subseteq SBSNNI$. Let $E \in P\_BNDC$. By definition of P_BNDC, for all E' reachable from E, $E' \in BNDC$ and then, by Proposition A.3, $E' \in BSNNI$. Hence, by Definition A.2, for all E' reachable from E, $E' \setminus Act_H \approx E'/Act_H$, i.e., $E \in SBNNI$.

The fact that $SBSNNI \subseteq P\_BNDC$ is demonstrated in the proof of Proposition 6 in [8]. □