Secure Contexts for Confidential Data[*]

Annalisa Bossi, Damiano Macedonio, Carla Piazza, Sabina Rossi
Dipartimento di Informatica - Università Ca' Foscari di Venezia
via Torino 155, 30172 Venezia, Italy
e-mail:{bossi,mace,piazza,srossi}@dsi.unive.it

Abstract

Information flow security in a multilevel system aims at guaranteeing that no high level information is revealed to low level users, even in the presence of any possible malicious process. This requirement could be too demanding when some knowledge about the environment (context) in which the process is going to run is available. To deal with these situations we introduce the notion of secure contexts for a class of processes. *This notion is parametric with respect to both the observation equivalence and the operation used to characterize the low level behavior of a process. We mainly analyze the cases of bisimulation and trace equivalence. We describe how to build secure contexts in these cases and we show that two well-known security properties, named* BNDC *and* NDC*, are just special instances of our general notion.*

1. Introduction

The problem of protecting confidential data in a multilevel system is one of the relevant issues in computer security. Information flow security aims at guaranteeing that no high level (confidential) information is revealed to users running at low levels [8, 14, 5, 17, 22, 20], even in the presence of any possible malicious process. An early attempt to formalize the absence of information flow was the concept of *noninterference* proposed in the seminal paper by Goguen and Meseguer [9], and further developed in [5, 6, 3, 11, 18, 21, 10]. Intuitively, to establish that information does not flow from high to low it is sufficient to establish that high behavior has no effect on what low level users can observe, i.e., the low level view of the system is independent of high behavior. A process which is secure with respect to this notion is thus secure whatever the surrounding high level environment is.

Our work starts from the observation that such a requirement could be too demanding when some knowledge about the environment (context) in which the process is going to run is available. Thus, we face the problem of generalizing the notion of noninterference to deal with specific classes of contexts. In our approach a context can perform both high and low level actions and can also incorporate possible attackers.

As an example consider a process representing a client of a bank using his cash card in an Automatic Teller Machine (ATM) to make a withdrawal from his account. When the card is inserted in the ATM the code of the card is read, then the client can write his PIN code, and if the PIN is correct he can ask for the cash. All the actions involved concern the exchange of confidential (high level) information between the client and the bank. We can formalize the process representing the client in front of the ATM as a CCS-like term, named $\textsc{client}$, of the form $\overline{\textsc{card}}.\overline{\textsc{pin}}.\textsc{cash}.\mathbf{0}$, where $\overline{\textsc{card}}$ and $\overline{\textsc{pin}}$ denote the client's output actions of giving the card and the pin codes, while $\textsc{cash}$ represents the input action of receiving the cash. All these actions are classified as high level actions. A *correct* ATM should read the codes, and if they are correct, it should give the cash to the client. Hence, leaving out the details concerning the checks of the codes, we represent an ATM as a context of the form $X|\textsc{card}.\textsc{pin}.\overline{\textsc{cash}}.\mathbf{0}$ where the variable X will be replaced by $\textsc{client}$ when the client will interact with the machine. Since all the data are protected, no (high) information is revealed to an external observer; hence we can assume that the above ATM context is secure for the process $\textsc{client}$.

Imagine now that a maintenance engineer puts a laptop inside the ATM. The laptop records all the card numbers and the PINs of the ATM's users. We can also imagine that once the confidential data have been captured the laptop send them to the bank so that the client receives the cash and does not suspect the fraud. Then $\textsc{laptop}$ is $\textsc{card}.\textsc{pin}.\overline{\textsc{record}}.\overline{\textsc{card}}.\overline{\textsc{pin}}.\mathbf{0}$, where $\textsc{record}$ is a non protected (low) output action. The counterfeit context is the parallel composition of the two components: the correct

[*]This work has been partially supported by MURST project "Modelli formali per la sicurezza" and the EU project MyThS (IST-2001-32617).

ATM and the LAPTOP. Clearly, this context is not secure for the process CLIENT. However, this does not mean that we give up using cards and ATMs. We just want to be sure to use them in secure contexts.

To deal with these situations we introduce the notion of *secure contexts for a class of processes*. This notion is parametric with respect to both an observation equivalence relation and an operation used to characterize the low level view of a process. In this paper we consider instances with weak bisimulation and trace equivalence as observation equivalence. We show how to build secure contexts and prove that the security properties known as *BNDC* and *NDC* (see [5]) are just special instances of our general security notion.

The paper is organized as follows. In Section 2 we recall the SPA language and its semantics. Section 3 introduces our definition of secure contexts for a class of processes. In Sections 4 and 5 we study two instances of our general definition through weak bisimulation and trace equivalence, respectively. Some conclusions are drawn Section 7.

2. Basic Notions

The *Security Process Algebra* (SPA) [5] is a variation of Milner's CCS [16], 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, i.e.: 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 special action τ which models internal computations, not visible outside the system; a complement function $\bar{\cdot} : \mathcal{L} \to \mathcal{L}$, such that $\bar{\bar{a}} = a$, for all $a \in \mathcal{L}$. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. Function $\bar{\cdot}$ is extended to Act by defining $\bar{\tau} = \tau$. The set of visible actions is partitioned into two sets, H and L, of high and low actions such that $\overline{H} = H$ and $\overline{L} = L$. The syntax of SPA *terms* is defined as follows:

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

where Z is a variable, $a \in Act$, $v \subseteq \mathcal{L}$, $f : Act \to Act$ is such that $f(\bar{\alpha}) = \overline{f(\alpha)}$, $f(\tau) = \tau$, $f(H) \subseteq H \cup \{\tau\}$, and $f(L) \subseteq L \cup \{\tau\}$.

We apply the standard notions of *free* and *bound* (occurrences of) variables in a SPA term. More precisely, all the occurrences of the variable Z in $recZ.T$ are *bound*; while Z is *free* in a term T if there is an occurrence of Z in T which is not bound.

Definition 2.1 A SPA *process* is a SPA term without free variables. We denote by $\mathcal{E}$ the set of all SPA processes, ranged over by $E, F, \ldots$, and by $\mathcal{E}_H$ the set of all high level processes, i.e., those constructed only using actions belonging to $H \cup \{\tau\}$.

The operational semantics of SPA processes is given in terms of *Labelled Transition Systems* (LTS, for short). In particular, the LTS $(\mathcal{E}, Act, \to)$, whose states are processes, is defined by structural induction as the least relation generated by the axioms and inference rules reported in Figure 1, where a is an action of Act, while l belongs to $\mathcal{L}$.

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 nondeterministic 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 executions are interleaved, possibly synchronized on complementary input/output actions, producing the silent 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; if in T there is at most the free variable Z, then $recZ.T[Z]$ is the recursive process which can perform all the actions of the process obtained by substituting $recZ.T[Z]$ to the placeholder Z in the context $T[Z]$.

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 $f_v(a) = a$ if $a \notin v$ and $f_v(a) = \tau$ if $a \in v$. In practice, E/v turns all actions in v into internal τ's.

A SPA term with free variables can be seen as an environment with holes (the free occurrences of its variables) in which other SPA terms can be inserted. The result of this substitution is still a SPA term, which could be a process. For instance, in the term $h.\mathbf{0}|(l.X + \tau.\mathbf{0})$ we can replace the variable X with the process $\bar{h}.\mathbf{0}$ obtaining the process $h.\mathbf{0}|(l.\bar{h}.\mathbf{0} + \tau.\mathbf{0})$; or we can replace X by the term $a.Y$ obtaining the term $h.\mathbf{0}|(l.a.Y + \tau.\mathbf{0})$. When we consider a SPA term as an environment we call it *context*.

Definition 2.2 A SPA *context*, ranged over by $C, D, \ldots$, is a SPA term in which free variables may occur.

We can also consider a context as a derived SPA *constructor*. In fact it can be used to build SPA terms from sets of SPA terms. Its arity is determined by the number of its free variables. For instance $X|X$ can be seen as a constructor of arity 1 which transforms any process E into the parallel composition with itself, $E|E$.

Given a context C, we use the notation $C[Y_1, \ldots, Y_n]$ to stress the fact that we are interested only in the free occurrences of the variables $Y_1, \ldots, Y_n$ in C. The term $C[T_1, \ldots, T_n]$ is obtained from $C[Y_1, \ldots, Y_n]$ by replacing all the free occurrences of $Y_1, \ldots, Y_n$ with the terms $T_1, \ldots, T_n$, respectively. For instance, we can write $C[X] \equiv h.\mathbf{0}|(l.X + \tau.\mathbf{0})$ or $D[X] \equiv (l.X + \tau.\mathbf{0})|Y$ or $C'[X] \equiv Y|h.\mathbf{0}$. Hence, the notation $C[\bar{h}.\mathbf{0}]$ stands for

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

Prefix $$\frac{-}{a.E \xrightarrow{a} E}$$

Sum $$\frac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'}$$

Parallel $$\frac{E_1 \xrightarrow{a} E_1'}{E_1|E_2 \xrightarrow{a} E_1'|E_2} \qquad \frac{E_2 \xrightarrow{a} E_2'}{E_1|E_2 \xrightarrow{a} E_1|E_2'} \qquad \frac{E_1 \xrightarrow{\ell} E_1' \quad E_2 \xrightarrow{\bar{\ell}} E_2'}{E_1|E_2 \xrightarrow{\tau} E_1'|E_2'}$$

Restriction $$\frac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v} \text{ if } a \notin v$$

Relabelling $$\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$$

Recursion $$\frac{T[recZ.T[Z]] \xrightarrow{a} E'}{recZ.T[Z] \xrightarrow{a} E'}$$

Figure 1. The operational rules for SPA

$h.\mathbf{0}|(l.\bar{h}.\mathbf{0} + \tau.\mathbf{0})$, while $D[\bar{h}.\mathbf{0}] \equiv (l.\bar{h}.\mathbf{0} + \tau.\mathbf{0})|Y$ and $C'[\bar{h}.\mathbf{0}] \equiv Y|h.\mathbf{0}$. Note that the notation $C[Y_1, \ldots, Y_n]$ implies neither that all the variables $Y_1, \ldots, Y_n$ occur free in the context nor that they include all the variables occurring free in the context. Note also that if W is a variable not occurring in $recZ.C[Z]$ and we replace all the occurrences of Z in $recZ.C[Z]$ by W we obtain the process $recW.C[W]$ (α-conversion) which is semantically equivalent to $recZ.C[Z]$. Nevertheless, the two terms $recZ.C[Z]$ and $recW.C[W]$ represents two different contexts.

The concept of *observation equivalence* is used to establish equalities among processes and it 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 $\mathcal{E}$ equating two processes when they are indistinguishable. In this paper we consider the relations named *weak bisimulation*, $\approx_B$, and *trace equivalence*, $\approx_T$.

Let us first introduce the following auxiliary notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \xrightarrow{a_1} \cdots \xrightarrow{a_n} E'$, then we write $E \xrightarrow{t} E'$ and we say that E' is *reachable* from E. We also write $E \overset{t}{\Longrightarrow} E'$ if $E(\xrightarrow{\tau})^* \xrightarrow{a_1} (\xrightarrow{\tau})^* \cdots (\xrightarrow{\tau})^* \xrightarrow{a_n} (\xrightarrow{\tau})^* E'$ where $(\xrightarrow{\tau})^*$ denotes a (possibly empty) sequence of τ labelled transitions. If $t \in Act^*$, then $\hat{t} \in \mathcal{L}^*$ is the sequence gained by deleting all occurrences of τ from t. As a consequence, $E \overset{\hat{a}}{\Longrightarrow} E'$ stands for $E \overset{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\overset{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\overset{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions).

The *weak bisimulation* relation [16] equates two processes if they are able to mutually simulate their behavior step by step. Weak bisimulation does not care about internal τ actions.

Definition 2.3 [Weak Bisimulation] A binary relation $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ over processes is a *weak bisimulation* if $(E, F) \in \mathcal{R}$ implies, for all $a \in Act$,

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

Two processes $E, F \in \mathcal{E}$ are *weakly bisimilar*, denoted by $E \approx_B F$, if there exists a weak bisimulation $\mathcal{R}$ containing the pair (E, F).

The relation $\approx_B$ is the largest weak bisimulation and it is an equivalence relation.

The *trace equivalence* relation equates two processes if they have the same sets of traces, again, without considering the τ actions.

Definition 2.4 [Trace Equivalence] For any process $E \in \mathcal{E}$ the set of traces $Tr(E)$ associated with E is defined by:

$Tr(E) = \{t \in \mathcal{L}^* \mid \exists E' \, E \overset{t}{\Longrightarrow} E'\}$. Two processes $E, F \in \mathcal{E}$ are *trace equivalent*, denoted by $E \approx_T F$, if $Tr(E) = Tr(F)$.

Trace equivalence is less demanding than weak bisimulation, hence if two processes are weakly bisimilar, then they are also trace equivalent.

Following [16] we extend binary relations on processes to contexts as follows.

Definition 2.5 [Relations on Contexts] Let $\mathcal{R}$ be a binary relation over processes, i.e., a subset of $\mathcal{E} \times \mathcal{E}$. Let C and D be two contexts and $\{Y_1, \ldots, Y_n\}$ be a set of variables which include all the free variables of C and D. We say that $C \, \mathcal{R} \, D$ if $C[E_1, \ldots, E_n] \;\, \mathcal{R} \;\, D[E_1, \ldots, E_n]$ for all set of processes $\{E_1, \ldots, E_n\}$.

In the case of weak bisimulation, applying the above definition we have that two contexts are weakly bisimilar if all the processes obtained by instantiating their variables are pairwise bisimilar. For instance, using our notation, the contexts $C[X] \equiv a.X + \tau.Y$ and $D[X] \equiv a.\tau.X + \tau.Y$ are weakly bisimilar since for all $E, F \in \mathcal{E}$ it holds $a.E + \tau.F \approx_B a.\tau.E + \tau.F$. Notice that not all the free variables of C and D were explicit in the notation $C[X]$ and $D[X]$. However, Definition 2.5 requires the instantiation of all their free variables.

3. Secure Contexts

In this section we introduce our notion of *secure contexts for a class of processes*. This notion is parametric with respect to an operation $\cdot_l$ used to characterize the low level behavior, E_l, of a process E, and an observation equivalence $\sim$ used to equate two processes. We denote by $\sim_l$ the relation $\sim$ on the low level views of processes, i.e., $E \sim_l F$ stands for $E_l \sim F_l$.

Definition 3.1 [Secure Contexts for a Class of Processes] Let $\sim$ and $\cdot_l$ be an observation equivalence relation and an operation on processes, respectively. Let $\mathcal{C}$ be a class of contexts, $\mathcal{P}$ be a class of processes, and X be a variable. The class $\mathcal{C}$ is *secure for the class* $\mathcal{P}$ *with respect to the variable* X if for all $C[X] \in \mathcal{C}$ and for all $E \in \mathcal{P}$,

$$C[E] \sim_l C[E_l].$$

In this definition the variable X is used to determine the "holes" in C which are intended to be filled in by E. Recall that X might not occur free in C. In this case C is trivially secure (by reflexivity of $\sim$). Moreover, in C there can be other free variables different from X. In this case we have to apply Definition 2.5 and instantiate the other free variables in all the possible ways.

Example 3.2 Let $\sim$ and $\cdot_l$ be an observation equivalence relation and an operation on processes, respectively. Let $\mathcal{P} = \{E\}$ and $\mathcal{C} = \{l.X + l.Y + h.Y\}$, with $l \in L$ and $h \in H$. To prove that $\mathcal{C}$ is secure for $\mathcal{P}$ with respect to the variable X we have to prove that for all $F \in \mathcal{E}$ it holds $l.E + l.F + h.F \sim_l l.E_l + l.F + h.F$. Similarly, to prove that $\mathcal{C}$ is secure for $\mathcal{P}$ with respect to the variable Y we have to prove that for all $F \in \mathcal{E}$ it holds $l.F + l.E + h.E \sim_l l.F + l.E_l + h.E_l$. The class $\mathcal{C}$ is trivially secure for $\mathcal{P}$ with respect to the variable Z, since for all $F, G \in \mathcal{E}$ it holds that $l.F + l.G + h.G \sim_l l.F + l.G + h.G$. □

In the rest of this paper when we say that $\mathcal{C}$ is secure for $\mathcal{P}$ we are implicitly referring to the variable X.

The intended meaning of our security definition is that a low level observer cannot distinguish the interactions between a process $E \in \mathcal{P}$ and a context $C \in \mathcal{C}$ from the interactions between the low level view E_l of E and C. If, accordingly with our intuition, E_l represents the low level behavior of E then our definition is clearly in the spirit of the *noninterference* schema proposed in [9].

Let us analyze the definition in the case in which only one process and one context are involved. The definition can be read from two points of view: security for the process and security for the context. On the one hand, if a context C is secure for a process E, then E can safely interact with C (security for the process), since C is not able to reveal to the low level users any high level information contained in E. In fact, it is revealed only the information that would be revealed by the interaction with E_l. On the other hand, if a context C is secure for a process E, then C can safely interact with E (security for the context). In fact, E is able to reveal the same information which could be revealed by E_l that cannot interact with the high level actions of C. In the introduction we gave a first example fitting with the first situation. Here we add two more examples to explain the two points of view.

Example 3.3 [Security for the Processes] Suppose that *Wholesaler ltd* is a wholesale company which does not sell its products directly to the final users but only to the shopkeepers. Thus the price of its products can be seen as a confidential data that only the *Wholesaler*'s customers (shopkeepers) are allowed to know. On the other hand the company advertises its products both to shopkeepers (high level) and to potential (low level) users. Consider a Java applet E downloadable from the site of *Wholesaler* ltd which should allow the shopkeepers to get confidential data like prices and the rest of the world to get a product list with generic information about the products. The applet opens a window with two buttons: the first button allows to read the product list, while the second one allows to read the price list, provided a password is inserted. Let PWD_SHOPKEEPER be the high level action representing the fact that E is waiting for a password from a shopkeeper before showing the price list. We assume that this is the only protection for the confidential data in E. The applet E can be represented by the following SPA process,

$$\text{PWD\_SHOPKEEPER}.\text{PRICES} + \text{PRODUCTS}$$

Wholesaler does not want the applet to be executed on a machine (context) which reveals some high level information (e.g., the price list) to non authorized users. Let us consider two possible contexts. Let C_1 be the machine of the high level user in which the password has been stored (in a cookie). Then C_1 can be represented by a term of the form

$$X|\overline{\text{PWD\_SHOPKEEPER}}.\mathbf{0}.$$

In this case high level information can be revealed: when a low level user interacts with $C_1[E]$, he (she) can read the price list. Hence, C_1 cannot be considered secure for E. Another more involved context is, for instance, a machine C_2 shared between high and low level users such that only high level users (shopkeepers) can read the price list, while low level ones can read the product list:

$$\text{PWD\_HIGH}.(X|\overline{\text{PWD\_SHOPKEEPER}}.\mathbf{0}) + \text{PWD\_LOW}.X.$$

In this case the flexibility of the context is obtained by splitting C_2 into two non-deterministic components: the first one manages the interaction with high level users and has in memory the shopkeeper's password; the second one interacts with low level users and does not provide any password. Note that if a high level user interacts with $C_2[E]$ by inserting the password PWD_HIGH, the PRICES component becomes accessible to low level observers. This can be seen as the possibility for the high level user to *downgrade* the level of the information stored in the price-list. Intuitively, the process E described here does not satisfy information flow security properties such as noninterference [17]. However, whenever downgrading is a high level user decision, it is reasonable to assume that the context C_2 is secure for E. □

Example 3.4 [Security for the Contexts] *Mr Earner* has on his own machine C some files containing the information about his investments. He would like to check whether they are profitable and, if they are not, to have some suggestions about how to change them. He installed on his machine a program which is able to check on the stock market through an Internet connection, reads his investments files and performs some computations to determine whether the investments are profitable or not. If the investments are going bad, the program checks again on the stock market, for better opportunities. The second check on the stock market is recommended since it allows to use the last quotations for computing suggestions (it is preferable not to use the cached stock market's quotations for this operation). Obviously *Mr Earner* does not want that someone knows if his investments are good or not. The machine of *Mr Earner* can be in one of the following states:

$$X|\overline{\text{GOOD}}.\mathbf{0} \qquad \text{or} \qquad X|\overline{\text{BAD}}.\text{SUGGESTIONS}.\mathbf{0}$$

which we assume to correctly represent the reality of his investments. In the first case *Mr Earner* investments are good and this fact can be revealed through the high level output $\overline{\text{GOOD}}$. In the second case *Mr Earner* investments are bad, hence after the high level output his machine is ready to have in input some suggestions through the high level input action SUGGESTIONS. *Mr Earner* wants both contexts be secure with respect to his investment program. Let us assume that *Mr Earner* investments are good, i.e., we consider the first context[2]. Let E_1 be the following program

$$\text{CHECK}.(\text{GOOD}.\mathbf{0} + \text{BAD}.\text{CHECK}.\overline{\text{SUGGESTIONS}}.\mathbf{0}),$$

where the only low level action is the input CHECK. By observing that E_1 has not checked a second time on the stock marked, a low level observer could be able to deduce that *Mr Earner*'s investments are good. Hence, in both cases the context representing *Mr Earner*'s machine is not secure with respect to E_1. However, it is secure with respect to the program E_2 of the form

$$\begin{aligned}\text{CHECK}.(\ &\text{GOOD}.\text{CHECK}.\mathbf{0} + \\ &\quad \text{BAD}.\text{CHECK}.\overline{\text{SUGGESTIONS}}.\mathbf{0} + \text{CHECK}.\mathbf{0}\ )\end{aligned}$$

which always performs a second check. Notice, that this behavior recalls the case of military radio transmissions. In order to avoid that someone knows when some information has been transmitted, every n instants a message is sent. Only one of the messages contains the real information.

Finally, if the market is "stable" and the elaboration of the information in *Mr Earner*'s file is "fast", the following program E_3 can be used

$$\text{CHECK}.(\text{GOOD}.\mathbf{0} + \text{BAD}.\overline{\text{SUGGESTIONS}}.\mathbf{0}).$$

It performs the low level input only once before analyzing the situation of the investments and gives its suggestions using the cached data. Also in this case, *Mr Earner*'s machine is secure with respect to this investment program E_3. □

When the class $\mathcal{C}$ has only one element C we say that C is secure for $\mathcal{P}$. Similarly, in the case in which $\mathcal{P}$ has only one element E we say that the class $\mathcal{C}$ is secure for the process E. If a context is secure for a class $\mathcal{P}$ of processes, then it is secure also for all the subclasses of $\mathcal{P}$. Analogously, if a class of contexts $\mathcal{C}$ is secure for a process E, then all the subclasses of $\mathcal{C}$ are secure for E. In the general case we obtain the following result.

[2]All the considerations which follow hold also for the second context.

Proposition 3.5 Let $\mathcal{C}_1 \subseteq \mathcal{C}_2$ be two classes of contexts, $\mathcal{P}_1 \subseteq \mathcal{P}_2$ be two classes of processes, and X be a variable. If $\mathcal{C}_2$ is secure for $\mathcal{P}_2$ with respect to X, then $\mathcal{C}_1$ is secure for $\mathcal{P}_1$ with respect to X.

Definition 3.1 introduces a general security notion. To analyze it more concretely it is necessary to instantiate the observation equivalence $\sim$ and the operation $\cdot_l$ defining the low level view of processes. A reasonable requirement to get useful instances is that of using a decidable equivalence and a computable operation.

In the next two sections we consider two instances of our framework. We study the properties of these instances and their connections with some security notions coming from the literature.

4. First Instance: Weak Bisimulation and Restriction

We analyze the properties of our security definition by instantiating the observation equivalence $\sim$ and the operation $\cdot_l$ as follows: $\sim$ is $\approx_B$ (weak bisimulation) and $\cdot_l$ is $\cdot \setminus H$ (restriction on high level actions). Using such an instance, a class of contexts $\mathcal{C}$ is secure for a class of processes $\mathcal{P}$ with respect to a variable X if for all $C[X] \in \mathcal{C}$ and for all $E \in \mathcal{P}$,

$$C[E] \setminus H \approx_B C[E \setminus H] \setminus H.$$

In the rest of this section we refer to this instance of our security property.

Example 4.1 Consider again Example 3.3 where confidential data are protected only by the password PWD_SHOPKEEPER. Assume that PRODUCTS and PRICES show the list of products and of prices to any (low or high) user asking for them. In SPA this behavior is obtained by creating two output actions for both the product and the price list, one for the low level users and the other for the high level ones.

$$\begin{aligned} \text{PRODUCTS} &= \overline{\text{PROD\_LIST\_H}}.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0} \\ \text{PRICES} &= \overline{\text{PRICE\_LIST\_H}}.\mathbf{0} + \overline{\text{PRICE\_LIST\_L}}.\mathbf{0}. \end{aligned}$$

$C_1[E] \setminus H \equiv \tau.\overline{\text{PRICE\_LIST\_L}}.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$ is not weakly bisimilar to $C_1[E \setminus H] \setminus H \equiv \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$. Indeed, a low level user interacting with $C_1[E]$ can read the price list, thus leaking confidential data. On the other hand, both $C_2[E] \setminus H$ and $C_2[E \setminus H] \setminus H$ are bisimilar to $\text{PWD\_LOW}.\overline{\text{PROD\_LIST\_L}}.\mathbf{0}$, according to the intuition that C_2 is secure for E. □

Example 4.2 In Example 3.4 we said that both the contexts representing *Mr Earner*'s machine are secure with respect to the second program E_2. Indeed, E_2 never reveals to low level users the situation of *Mr Earner*'s investments, since a second check on the marked is performed in any case. For instance, using the first context of Example 3.4 we obtain that $C[E_2] \setminus H \equiv \text{CHECK}.(\tau.\text{CHECK}.\mathbf{0} + \text{CHECK}.\mathbf{0})$ is weakly bisimilar to $C[E_2 \setminus H] \setminus H \equiv \text{CHECK}.\text{CHECK}.\mathbf{0}$, hence the security property holds.

The third program E_3 of Example 3.4 satisfies that $C[E_3] \setminus H \approx_B C[E_3 \setminus H] \setminus H$ for both the contexts, as it can be easily checked. □

Using this first instance we find an interesting connection between our security definition and the security property known as *BNDC* and proposed by Focardi and Gorrieri in [4]. The security property *BNDC* is based on the idea of checking the system against all high level potential interactions, representing every possible high level malicious program. In particular, a process E is *BNDC* if for every high level process Π a low level user cannot distinguish E from $(E|\Pi)$, i.e., if Π cannot interfere with the low level execution of E.

Definition 4.3 [BNDC] Let $E \in \mathcal{E}$. $E \in BNDC$ if for all $\Pi \in \mathcal{E}_H$,

$$E \setminus H \approx_B (E|\Pi) \setminus H.$$

The following lemma states that the set of contexts of the form $X|\Pi$ with $\Pi \in \mathcal{E}_H$ characterizes the class of *BNDC* processes.

Lemma 4.4 Let $E \in \mathcal{E}$. $E \in BNDC$ iff $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ for all contexts $C[X] \equiv X|\Pi$ with $\Pi \in \mathcal{E}_H$.

PROOF. See Appendix. □

Example 4.5 The process E in Example 3.3 is not a *BNDC* process. In fact, the context $X|\overline{\text{PWD\_SHOPKEEPER}}.\mathbf{0}$ is a context of the form $X|\Pi$ with $\Pi \in \mathcal{E}_H$ and it is not secure for E, hence by Lemma 4.4 we obtain that E is not *BNDC*. However, as shown in Example 4.1, there are complex contexts in which E can be safely executed.

Both processes E_2 and E_3 of Example 3.4 can be proved to be *BNDC* process. □

In Subsection 4.1 we identify two classes of contexts which are secure for all the processes. Then, in Subsection 4.2 we concentrate on classes of processes characterized by some security notions (basically we will consider subclasses of *BNDC*) and analyze whether there exist larger classes of secure contexts for them.

4.1. $\approx_B$ Instance: Secure Contexts for a generic class $\mathcal{P}$

Next theorem provides a compositionality result for secure contexts with respect to any class of processes.

Theorem 4.6 Let $\mathcal{P}$ be a class of processes. Let $\mathcal{C}$ be the class of contexts containing all $F \in \mathcal{E}$; all variables; all contexts of the form $\sum_{l_i \in L} l_i.C_i + \sum_{h_j \in H} h_j.D_j$, with the C_i's secure for $\mathcal{P}$ with respect to X; all contexts $C \setminus v$ and $C[f]$ with C secure for $\mathcal{P}$ with respect to X. Then $\mathcal{C}$ is secure for $\mathcal{P}$ with respect to X.

PROOF. See Appendix. □

Notice that it does not hold that if C and D are secure for $\mathcal{P}$, then $C|D$ is secure for $\mathcal{P}$. This is a consequence of the fact that we do not know anything about the class $\mathcal{P}$.

Example 4.7 Consider the class $\mathcal{P} = \{E\}$ where $E \equiv h.l.\mathbf{0} + \bar{h}.\mathbf{0}$. The context X is secure for $\mathcal{P}$ (see Theorem 4.6), but the context $X|X$ is not secure for $\mathcal{P}$. □

Observe that Theorem 4.6 does not provide a decidability result. For instance, if we know that C is secure for $\mathcal{P}$, then we can deduce that $C \setminus v$ is secure for $\mathcal{P}$, but, in general, we cannot use Theorem 4.6 to prove that $C \in \mathcal{C}$ and thus it is secure for $\mathcal{P}$.

Hereafter we characterize a decidable class of contexts which are secure for all the processes (i.e., for a generic class $\mathcal{P}$). Obviously we want the class to be as large as possible. In order to obtain the decidability of the class we require a compositionality structure, i.e., contexts are build only using sub-contexts which belong to the class. In order to ensure security we do not use the parallel composition when the context is not a closed term (see Example 4.7).

Definition 4.8 [The Class $\mathcal{C}_s$] Let $\mathcal{C}_s$ be the class of contexts which contains all the SPA processes, all the variables, and is closed with respect to the following constructors: $\sum_{i\in I} a_i.Y_i$ (with $a_i \in Act$), $Y \setminus v$, $Y[f]$, $recZ.Y$.

Notice that if $C[Y], D \in \mathcal{C}_s$, then we have $C[D] \in \mathcal{C}_s$.

The class $\mathcal{C}_s$ is decidable, in fact it is easy to define a proof system whose proofs correspond exactly to the constructions of the contexts in $\mathcal{C}_s$.

Example 4.9 The contexts X, Y and Z belong to $\mathcal{C}_s$. Hence, by using the constructor $a.Y_1 + b.Y_2 + c.Y_2$, the context $a.X + b.Y + c.Z$ belongs to $\mathcal{C}_s$, and then, by using the $recY.W$ constructor, the context $recY.(a.X + b.Y + c.Z)$ is in $\mathcal{C}_s$. □

All the contexts in $\mathcal{C}_s$ are secure for all the processes, as it is stated by the next theorem. The following lemmas are used in its proof.

Lemma 4.10 The relation $\approx_B$ is a congruence in the class $\mathcal{C}_s$ with respect to its constructors.

PROOF. See Appendix. □

Lemma 4.11 Let $\mathcal{P}$ be a class of processes and $C[X] \in \mathcal{C}_s$ be secure for $\mathcal{P}$ with respect to X. The context $recY.C[X]$ is secure for $\mathcal{P}$ with respect to X.

PROOF. See Appendix. □

Theorem 4.12 Let $\mathcal{P}$ be a class of processes and X be a variable. If $C \in \mathcal{C}_s$, then C is secure for $\mathcal{P}$ with respect to X.

PROOF. The proof follows by induction on the structure of the context C.

- $C \in \mathcal{E}$. We have already proved in Theorem 4.6, that C is secure for $\mathcal{P}$.
- $C \equiv Y$. Again, this has been proved in Theorem 4.6.
- $C \equiv \sum_{i\in I} a_i.C_i$. By induction on the C_i's and by Lemma 4.10 we have the thesis.
- $C \equiv C_1 \setminus v$. By induction on C_1 and applying Lemma 4.10 we obtain the thesis.
- $C \equiv C_1[f]$. Again, by induction on C_1 and Lemma 4.10 we get the thesis.
- $C \equiv recY.C_1$. By induction on C_1 and Lemma 4.11 we have the thesis.

□

Example 4.13 Let C be a machine shared between one low level user and one high level user. When one of the two users is logged, the machine cannot be used by the other one. The logged user can execute his program or a new program which has been downloaded from the web. The programs of both the users always terminate and at the end of their executions the other user can take the control. Let PWD_HIGH be high level action representing the input of the high level user password. Moreover, let CALL_PROG_H be the high level call to the program and $\overline{\text{EX\_PROG\_H}}$ its execution. Finally, let CALL_WEB_H be the high level call to the program downloaded from the web. All the low level actions are similarly defined. Hence, C has the form

$$\begin{array}{l} recY.(\ \text{PWD\_HIGH}.(\ \text{CALL\_PROG\_H}.\overline{\text{EX\_PROG\_H}}.Y \\ \qquad\qquad\qquad\qquad + \text{CALL\_WEB\_H}.X\ ) \\ \qquad + \text{PWD\_LOW}.(\ \text{CALL\_PROG\_L}.\overline{\text{EX\_PROG\_L}}.Y \\ \qquad\qquad\qquad\qquad + \text{CALL\_WEB\_L}.X\ )) \end{array}$$

Since C belongs to $\mathcal{C}_s$, C is secure for the program coming from the web with respect to X. □

As shown in Example 4.7, without assumptions on the class $\mathcal{P}$ the contexts built using the parallel operator cannot be considered secure. However, as seen in the previous examples most contexts involve the parallel operator, since it is at the core of the exchange of information between processes and contexts. For this reason in the next subsection we concentrate on classes of processes for which we prove that some contexts involving the parallel operator are secure.

4.2. $\approx_B$ Instance: Secure Contexts for sub-classes of *BNDC*

As stated in Lemma 4.4 some particular contexts built using the parallel operator are secure for the class *BNDC*. Unfortunately, the decidability of *BNDC* is still an open problem, and for this reason many sufficient conditions for *BNDC* have been introduced and studied in the literature (see [5, 7, 1]). In particular, in [1] three of these sufficient conditions have been considered and it has been shown that they can be parametrically characterized with respect to a suitable bisimulation relation. In virtue of Proposition 3.5, all the contexts which are secure for the largest of these three classes, that is the one named *P_BNDC*, are secure also for the other two classes. *P_BNDC* is nothing but the persistent version of *BNDC*. The persistence of *P_BNDC* has been proved to be fundamental to deal with dynamic contexts (see [7]).

Definition 4.14 [P_BNDC] Let $E \in \mathcal{E}$. $E \in P\_BNDC$ if $E' \in BNDC$ for all E' reachable from E.

In order to obtain that the parallel composition $C|D$ of secure contexts is still a secure context we need to be able to exchange the parallel operator with the restriction one, i.e., knowing that $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ and $D[E] \setminus H \approx_B D[E \setminus H] \setminus H$ we want to obtain that $(C[E]|D[E]) \setminus H \approx_B (C[E \setminus H]|D[E \setminus H]) \setminus H$. Such property holds for *P_BNDC* processes as shown by the following lemma.

Lemma 4.15 Let $E, F, G, K \in P\_BNDC$. If $E \setminus H \approx_B F \setminus H$ and $G \setminus H \approx_B K \setminus H$, then $(E|G) \setminus H \approx_B (F|K) \setminus H$.

PROOF. See Appendix. □

The previous lemma suggests that if we restrict to contexts mapping *P_BNDC* processes into *P_BNDC* processes we obtain that the parallel composition of secure contexts is secure. In this way we obtain the crucial result on compositionality which was missed in the previous section.

The following definition will be used also in the next section.

Definition 4.16 [$\mathcal{P}$-contexts] Let $\mathcal{P}$ be a class of processes and $C[X, Y_1, \ldots, Y_n]$ be a context whose free variables are in $\{X, Y_1, \ldots, Y_n\}$. $C[X, Y_1, \ldots, Y_n]$ is said to be a $\mathcal{P}$-*context with respect to* X if for all $E \in \mathcal{P}$ and for all $F_1, \ldots, F_n \in \mathcal{E}$ it holds that $C[E, F_1, \ldots, F_n] \in \mathcal{P}$.

Definition 4.17 [$\mathcal{P}$-secure contexts] Let $\mathcal{P}$ be a class of processes. A context $C[X]$ is said to be $\mathcal{P}$-*secure with respect to* X if it is a $\mathcal{P}$-context with respect to X and it is secure for $\mathcal{P}$ with respect to X.

Theorem 4.18 Let C and D be two contexts which are P_BNDC-secure with respect to X. The context $C|D$ is P_BNDC-secure with respect to X.

PROOF. The fact that $C|D$ is a P_BNDC-context follows from the fact that if two processes are P_BNDC, then their parallel composition is P_BNDC.

We prove that $C|D$ is secure for P_BNDC. If $E \in P\_BNDC$, then by hypothesis we have $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ and $D[E] \setminus H \approx_B D[E \setminus H] \setminus H$. Moreover, since $E \setminus H$ is always P_BNDC we have that $C[E], C[E \setminus H], D[E], D[E \setminus H]$ are P_BNDC. Hence, by applying Lemma 4.15 to the four processes we get the thesis. □

Notice that we can apply the theorem more than once, thus obtaining contexts which involve more parallel operators mixed with other operators.

From Proposition 3.5 we have that the contexts which can be proved to be secure using Theorem 4.18 are secure also for the two subclasses of P_BNDC named $SBNDC$ (see [5]) and CP_BNDC (see [1]), respectively.

Example 4.19 Consider the programs E_2 and E_3 of Example 3.4. They are P_BNDC, hence by applying Theorem 4.18 we immediately get that the two contexts of Example 3.4 are secure for these processes. □

Example 4.20 Let $\mathrm{END} \in \mathcal{L}$ be an action and E be a P_BNDC process in which neither END nor $\overline{\mathrm{END}}$ occur. Let $\mathcal{P}_{\mathrm{END}}$ be a class of P_BNDC processes whose termination is announced by the execution of an END action. Consider the context C defined as

$$(X|\overline{\mathrm{END}}.E) \setminus \{\mathrm{END}\}.$$

When in C we replace the variable X with a process F taken from $\mathcal{P}_{\mathrm{END}}$ we obtain that F is executed and then E is executed, i.e., we obtain a context which behaves like a sequential operator. From Theorem 4.18 and Proposition 3.5, we have that $X|\overline{\mathrm{END}}.E$ is secure for $\mathcal{P}_{\mathrm{END}}$. Hence, from Theorem 4.6, we obtain that C is secure for $\mathcal{P}_{\mathrm{END}}$. □

Theorem 4.18 does not provide a decidability result. In fact, to check that a context is a P_BNDC-context, in general, it is necessary to check that an infinite number of processes are in P_BNDC. The following definition characterizes a decidable class of contexts which are P_BNDC-contexts.

Definition 4.21 [The Class $\mathcal{C}_p$] Let $\mathcal{C}_p$ be the class of contexts which contains all the P_BNDC processes, the variable X, $Y \setminus H$ and Y/H for every variable Y, and is closed with respect to the following constructors: $Y|Z$, $Y \setminus v$, $Y[f]$, $\sum_{i\in I} l_i.Z_i + \sum_{j\in J}(h_j.Y_j + \tau.Y_j)$, where $l_i \in L$ and $h_j \in H$.

Example 4.22 The contexts X and $W \setminus H$ belong to $\mathcal{C}_p$. Hence, by using the constructor $l.Z_1 + h.Y_1 + \tau.Y_1$, the context $l.(W \setminus H) + h.X + \tau.X$ belongs to $\mathcal{C}_p$. □

Theorem 4.23 If $C[X] \in \mathcal{C}_p$ then $C[X]$ is P_BNDC-secure with respect to X.

PROOF. First we prove that all the contexts in $\mathcal{C}_p$ are P_BNDC-contexts. This is immediate by induction on the structure of the context. In particular, the case of the non deterministic choice can be proved using the unwinding characterization of *P_BNDC* presented in [2], while the case of the parallel operator is a consequence of the fact that the parallel composition of *P_BNDC* processes is *P_BNDC* (see [7]).

Now we prove that all the contexts in $\mathcal{C}_p$ are secure for P_BNDC. This is immediate by induction on the structure of the contexts. The basic steps are trivial. All inductive steps follow by Theorem 4.6 except the parallel case, which follows from Lemma 4.15. □

5. Second Instance: Trace Equivalence and Restriction

Sometimes weak bisimulation is too demanding since in some cases processes which are not weakly bisimilar can be considered equivalent.

Example 5.1 Consider again the process of Example 3.3. *Wholesaler* ltd could imagine that people usually set cookies. Hence, it could decide to change the applet in the following way: if the password is inserted, then the price list is given, but as an encrypted file. The high level user has to use another program to decrypt the file and this program does not allow to store the decryption key. In this case the price list is given in output only through a high level action and the process E becomes

$$\mathrm{PWD\_SHOPKEEPER}.\overline{\mathrm{PRICE\_LIST\_H}}.\mathbf{0} \\ + (\overline{\mathrm{PROD\_LIST\_H}}.\mathbf{0} + \overline{\mathrm{PROD\_LIST\_L}}.\mathbf{0}).$$

If we consider the context C_1, that is $X|\overline{\mathrm{PWD\_SHOPKEEPER}}.\mathbf{0}$, we have $C_1[E] \setminus H \equiv \tau.\mathbf{0} + \overline{\mathrm{PROD\_LIST\_L}}.\mathbf{0}$ is not weakly bisimilar to $C_1[E \setminus H] \setminus H \equiv \overline{\mathrm{PROD\_LIST\_L}}.\mathbf{0}$. However, the low level user cannot read the price list using this context. He can only infer whether a high level user has used the applet to read the price list. Since everybody knows that there exists a price list (and thus its existence is not a secret), in this case the use of bisimulation seems too restrictive. Trace equivalence could be the right choice. □

In this section we consider the following instance of our security definition: $\sim$ is $\approx_T$ (trace equivalence) and $\cdot_l$ is $\cdot \setminus H$ (restriction on high level actions). In this case a class of contexts $\mathcal{C}$ is secure for a class of processes $\mathcal{P}$ with respect to X if for all $C[X] \in \mathcal{C}$ and for all $E \in \mathcal{P}$,

$$C[E] \setminus H \approx_T C[E \setminus H] \setminus H.$$

In the rest of this section we refer to this instance of our security property.

Example 5.2 Consider the context C_1 and the process E of Example 5.1. Using the above instance of our security notion, C_1 is secure for E with respect to X. □

Let us consider the security property known as *NDC* (see [5]) which is defined similarly to *BNDC*, but using trace equivalence instead of weak bisimulation.

Definition 5.3 [NDC] Let $E \in \mathcal{E}$. $E \in NDC$ if for all $\Pi \in \mathcal{E}_H$,

$$E \setminus H \approx_T (E|\Pi) \setminus H.$$

The *NDC* security property is decidable as it immediately follows from the following characterization, whose proof can be found in [5].

Lemma 5.4 Let $E \in \mathcal{E}$. $E \in NDC$ iff $E/H \approx_T E \setminus H$.

As in the case of *BNDC*, it is possible to prove that all the contexts of the form $X|\Pi$ with $\Pi \in \mathcal{E}_H$ are secure for *NDC* processes.

Lemma 5.5 Let $E \in \mathcal{E}$. $E \in NDC$ iff $C[E] \setminus H \approx_T C[E \setminus H] \setminus H$ for all contexts $C[X] \equiv X|\Pi$ with $\Pi \in \mathcal{E}_H$.

PROOF. See Appendix. □

In the next subsection we study contexts which are secure, using this second instance, for all the processes. Then in Subsection 5.2 we concentrate on the contexts secure for the class of *NDC* processes.

5.1. $\approx_T$ Instance: Secure Contexts for a generic class $\mathcal{P}$

Since trace equivalence is less demanding than weak bisimulation we immediately obtain that the contexts which were secure in the previous section are secure also in this section.

Theorem 5.6 Let $\mathcal{C}$ be a class of contexts and $\mathcal{P}$ be a class of processes. If $C[E]\backslash H \approx_B C[E\backslash H]\backslash H$ for all $C[X] \in \mathcal{C}$ and for all $E \in \mathcal{P}$, then $C[E] \setminus H \approx_T C[E \setminus H] \setminus H$ for all $C[X] \in \mathcal{C}$ and for all $E \in \mathcal{P}$.

PROOF. Immediate consequence of the fact that if $E \approx_B F$ then $E \approx_T F$, for all $E, F \in \mathcal{E}$. □

This means that the class of contexts of Theorem 4.6 and the class $\mathcal{C}_s$ are secure for a generic class $\mathcal{P}$ of processes also with the second instance of our definition. Next theorem shows that we can enlarge the class of secure contexts for any $\mathcal{P}$.

Theorem 5.7 Let $\mathcal{P}$ be a class of processes and X be a variable. A context of the form $\sum_{i\in I} C_i + \sum_{h_j\in H} h_j.D_j$ is secure for $\mathcal{P}$ with respect to X if C_i is secure for $\mathcal{P}$ with respect to X for all $i \in I$.

PROOF. See Appendix. □

Notice that, also in this case it does not hold that if C and D are secure for $\mathcal{P}$, then $C|D$ is secure for $\mathcal{P}$. The contexts and the process presented in Example 4.7 witness this fact.

5.2. $\approx_T$ Instance: Secure Contexts for NDC processes

Here we rediscover the analogous of the results proved in Subsection 4.2 for *P_BNDC* processes, in the case of *NDC* processes. In particular, the following lemma is the correspondent of Lemma 4.15.

Lemma 5.8 Let $E, F, G, K \in NDC$. If $E \setminus H \approx_T F \setminus H$ and $G \setminus H \approx_T K \setminus H$, then $(E|G) \setminus H \approx_T (F|K) \setminus H$.

PROOF. See Appendix. □

This allows us to obtain the following result which states that contexts obtained using the parallel operator are secure for *NDC* processes when the two contexts which are put in parallel are secure and map *NDC* processes into *NDC* processes.

Definition 5.9 A context $C[X]$ is said to be *NDC-secure with respect to* X if it is a NDC-context with respect to X and it is secure for NDC with respect to X.

Theorem 5.10 Let C and D be two contexts which are NDC-secure with respect to X. The context $C|D$ is NDC-secure with respect to X.

PROOF. The fact that $C|D$ is a NDC-context follows from the fact that if two processes are NDC, then their parallel composition is NDC.

We prove that $C|D$ is secure for NDC. If $E \in NDC$, then by hypothesis we have $C[E] \setminus H \approx_T C[E \setminus H] \setminus H$ and $D[E] \setminus H \approx_T D[E \setminus H] \setminus H$. Moreover, since $E \setminus H$ is always NDC we have that $C[E], C[E \setminus H], D[E], D[E \setminus H]$ are NDC. Hence, by applying Lemma 5.8 to these four processes we get the thesis. □

Theorem 5.10 does not provide a decidability result. In the following definition we characterize a decidable class of NDC-contexts, which is the analogous of the class $\mathcal{C}_p$ of Definition 4.21.

Definition 5.11 [The Class $\mathcal{C}_n$] Let $\mathcal{C}_n$ be the class of contexts which contains all the NDC processes, the variable X, $Y \setminus H$ and Y/H for every variable Y, and is closed with respect to the following constructors: $l.Y$ with $l \in L$, $Y|Z$, $Y \setminus v$, $Y[f]$, $Y + Z$, $h.Y + \tau.Y$ with $h \in H$.

Theorem 5.12 If $C[X] \in \mathcal{C}_n$ then $C[X]$ is NDC-secure with respect to X.

PROOF. First we prove that all the contexts in $\mathcal{C}_n$ are NDC-contexts. This is immediate by induction on the structure of the context. In particular, we use the fact that trace equivalence is a congruence with respect to non deterministic choice, the fact that if $E, F \in NDC$ then $E|F, E \setminus H \in NDC$ (see [4]).

Now we prove that all the contexts in $\mathcal{C}_n$ are secure for NDC. This is immediate by induction on the structure of the context. The basic steps are trivial. As weak bisimulation implies trace equivalence, all the inductive steps follow by by Theorem 4.6 except cases of parallel and nondeterministic choice. The parallel step follows by Lemma 5.8. Finally, let $C[X]$ and $D[X]$ be secure for NDC, i.e. $Tr(C[E] \setminus H) = Tr(C[E \setminus H] \setminus H)$ and $Tr(D[E]\setminus H) = Tr(D[E\setminus H]\setminus H)$ for all $E \in NDC$, then $Tr((C[E]+D[E])\setminus H) = Tr((C[E]\setminus H)+(D[E]\setminus H)) = Tr(C[E] \setminus H) \cup Tr(D[E] \setminus H) = Tr(C[E \setminus H] \setminus H) \cup Tr(D[E \setminus H] \setminus H) = Tr(C[E \setminus H] + D[E \setminus H])$ for all $E \in NDC$, so we conclude that $C[X] + D[X]$ is secure for for NDC. □

6. Related Works

Since the seminal work by Goguen and Meseguer [9], noninterference plays a central role in the formalization of the notion of confidentiality. Nevertheless, many authors notice that it is too demanding when dealing with practical applications. In [18], Ryan and Schneider notice that no real policy ever calls for total absence of information flow over any channel, and that in any case this would not be achievable. Thus, they point out the need of investigating generalizations of the notion of noninterference to allow for partial or conditional flows and introduce a general definition of noninterference. They also discuss how such a generalization could be appropriate to deal with realistic practical situations, e.g., with policies that allow for automatic downgrading of certain statistical information from a database. Our definition follows the spirit of [18] and generalizes the formalization presented in that paper by allowing the use of more structured contexts and not considering only trace-based equivalences.

In [12], Martinelli observes that security properties can be naturally described as properties of open systems, i.e., systems which may have unspecified components. These may be used to represent a hostile intruder whose behavior cannot be predicted or a malicious system component. The verification mechanism proposed in [12] consists of checking that, for any instance of the unknown component, the resulting system satisfies a property expressed in a formula of a suitable temporal logic. In order to make decidable the verification problem, Martinelli does not consider constructs for modelling recursion.

Secure contexts are also studied by Sabelfeld and Mantel in [19] where they propose a timing-sensitive security definition for programs in a simple multi-threaded language. Sabelfeld and Mantel give a syntactic characterization of a class of contexts in their language which preserve security, i.e., they are secure whenever one substitutes holes with secure programs. This, in a sense, corresponds to our general definition of $\mathcal{P}$-secure contexts. In particular, their definition of secure contexts is based on a "hook-up" (compositionality) property [13] of their notion of security. That is contexts just reflect the compositionality property of their security notion. Actually the compositionality of security properties is a fundamental issue in the incremental definition of secure systems (see [15, 23, 21]). In this work we point out a strong relation between the compositionality properties of a class $\mathcal{P}$ of processes and the compositionality properties of $\mathcal{P}$-secure contexts (see Theorem 4.18). Moreover, we can use our definition of secure contexts to identify new classes of secure processes.

7. Conclusions

We presented a generalization of the notion of noninterference which is more flexible than the one introduced by Goguen and Meseguer and its subsequent versions. The flexibility is a consequence of the fact that our notion is parametric with respect to a class of contexts and thus not limited to contexts of the form $X|\Pi$, with $\Pi \in \mathcal{E}_H$.

On the one hand our notion can be used to restrict the set of possible attackers: e.g., when it is not reasonable to assume that an attacker has the ability to perform any high level action. This seems reasonable in practical applications: for instance, when we assume that no all high level passwords can be guessed.

On the other hand our notion allows us to enlarge the set of possible attackers, since SPA operators can be freely combined in the context construction.

Moreover, the possibility of using low level actions in building contexts is useful when we reverse the point of view, i.e., when we are interested in the security of the contexts with respect to a class of processes.

We studied two instances of our general definition with weak bisimulation and trace equivalence. We characterized decidable classes of secure contexts for *P_BNDC* and *NDC* processes.

An interesting future issue could be the reformulation of our security property in richer languages (e.g., π-calculus).

Acknowledgements

We would like to thank Francesco Dalla Libera for the helpful discussions and the anonymous reviewers for their comments and suggestions.

References

[1] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Bisimulation and Unwinding for Verifying Possibilistic Security Properties. In L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay, editors, *Proc. of Int. Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'03)*, volume 2575 of *LNCS*, pages 223–237. Springer-Verlag, 2003.

[2] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. A Proof System for Information Flow Security. In M. Leuschel, editor, *Logic Based Program Development and Transformation*, volume 2664 of *LNCS*. Springer-Verlag, 2003. To appear.

[3] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In M. Agrawal and A. Seth, editors, *Proc. of Int. Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'02)*, volume 2556 of *LNCS*, pages 71–84. Springer-Verlag, 2002.

[4] R. Focardi and R. Gorrieri. A Classification of Security Properties for Process Algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[5] 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*, pages 331–396. Springer-Verlag, 2001.

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

[7] R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 307–319. IEEE Computer Society Press, 2002.

[8] S. N. Foley. A Universal Theory of Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 116–122. IEEE Computer Society Press, 1987.

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

[10] M. Hennessy and J. Riely. Information Flow vs. Resource Access in the Asynchronous Pi-calculus. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 24(5):566–591, 2002.

[11] H. Mantel. Possibilistic Definitions of Security - An Assembly Kit -. In *Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 185–199. IEEE Computer Society Press, 2000.

[12] F. Martinelli. Analysis of Security Protocols as *open* Systems. *Theoretical Computer Science*, 290(1):1057–1106, 2003.

[13] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 161–166. IEEE Computer Society Press, 1987.

[14] J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 180–187. IEEE Computer Society Press, 1990.

[15] J. McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.

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

[17] C. O'Halloran. A Calculus of Information Flow. In *Proc. of the European Symposium on Research in Security and Privacy*, pages 180–187. AFCET, 1990.

[18] P. Ryan and S. Schneider. Process Algebra and Non-Interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.

[19] A. Sabelfeld and H. Mantel. Static Confidentiality Enforcement for Distributed Programs. In M. Hermenegildo and G. Puebla, editors, *Proceedings of the 9th International Static Analysis Symposium, Madrid, Spain, September 17-20*, number 2477 in LNCS, pages 376–394. Springer-Verlag, 2002.

[20] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on Selected Areas in Communication*, 21(1):5–19, 2003.

[21] A. Sabelfeld 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.

[22] G. Smith and D. M. Volpano. Secure Information Flow in a Multi-threaded Imperative Language. In *Proc. of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'98)*, pages 355–364. ACM Press, 1998.

[23] A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 74–102. IEEE Computer Society Press, 1997.

A. Proofs

Proof of Lemma 4.4. ($\Rightarrow$) If $E \in BNDC$, then $(E|\Pi) \setminus H \approx_B E \setminus H$. Moreover, $E \setminus H$ is always in $BNDC$ and $E \setminus H \setminus H \approx_B E \setminus H$, hence $(E \setminus H|\Pi) \approx_B E \setminus H$. So by transitivity of $\approx_B$, we obtain that $(E|\Pi) \setminus H \approx_B (E \setminus H|\Pi) \setminus H$.

($\Leftarrow$) Since $E \setminus H$ is always in $BNDC$ and $E \setminus H \setminus H \approx_B E \setminus H$, we obtain $(E|\Pi) \setminus H \approx_B (E \setminus H|\Pi) \setminus H \approx_B E \setminus H$. $\square$

Proof of Theorem 4.6.
- Since $F \setminus H \approx_B F \setminus H$ for each $F \in \mathcal{E}$, F is secure for $\mathcal{P}$.
- Since $E \setminus H \approx_B E \setminus H \setminus H$ for all $E \in \mathcal{E}$ any variable Y is secure for $\mathcal{P}$.
- Let $C[X] \equiv \sum_{l_i \in L} l_i.C_i + \sum_{h_j \in H} h_j.D_j$, with C_i secure for $\mathcal{P}$ for all i. We prove that $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ for all $E \in \mathcal{P}$. If $C[E] \setminus H \xrightarrow{a} C'$, then $a \in L$. Hence, there exists i such that $a = l_i$ and $C' \equiv C_i[E] \setminus H$. So we have that $C[E \setminus H] \setminus H \xrightarrow{a} C_i[E \setminus H] \setminus H$, and since $C_i[X]$ is secure for $\mathcal{P}$ it holds that $C_i[E] \setminus H \approx_B C_i[E \setminus H] \setminus H$. Similarly, if $C[E \setminus H] \setminus H \xrightarrow{a} C'$, then there exists i such that $a = l_i$ and $C' \equiv C_i[E \setminus H] \setminus H$. Hence, since $C_i[X]$ is secure for $\mathcal{P}$ we obtain that $C[E] \setminus H \xrightarrow{a} C_i[E] \setminus H$ with $C_i[E] \setminus H \approx_B C[E \setminus H] \setminus H$.
- Let $E \in \mathcal{P}$. From $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ we obtain $C[E] \setminus H \setminus v \approx_B C[E \setminus H] \setminus H \setminus v$, hence $C[E] \setminus v \setminus H \approx_B C[E \setminus H] \setminus v \setminus H$.
- Let $E \in \mathcal{P}$. We prove that $C[E][f] \setminus H \approx_B C[E \setminus H][f] \setminus H$, where f maps high actions in $H \cup \{\tau\}$ and low actions in $L \cup \{\tau\}$. If $C[E][f] \setminus H \xrightarrow{a} C'$, then $C' \equiv C''[f]$ and there exists b such that $f(b) = a$ and $C[E] \setminus H \xrightarrow{b} C''$. Hence, $C[E \setminus H] \setminus H \overset{\hat{b}}{\Rightarrow} C'''$ with $C'' \approx_B C'''$. So we obtain that $C[E \setminus H][f] \setminus H \overset{\hat{a}}{\Rightarrow} C'''[f]$ with $C''[f] \approx_B C'''[f]$. $\square$

Proof of Lemma 4.10. The only non trivial case is the "Recursion". Given $C, D \in \mathcal{C}_s$ with $C \approx_B D$ we have to prove that $recY.C \approx_B recY.D$. Without loss of generality we can assume that $C[Y]$ and $D[Y]$ have at most the single free variable Y. The general case follows from Definition 2.5. In fact, suppose that $C[Y, Y_1 \dots Y_n] \approx_B D[Y, Y_1 \dots Y_n]$, then for any choice of $E_1 \dots E_n \in \mathcal{E}$ we have $C[Y, E_1 \dots E_n] \approx_B D[Y, E_1 \dots E_n]$, and thus $recY.C[Y, E_1 \dots E_n] \approx_B recY.D[Y, E_1 \dots E_n]$; therefore $recY.C[Y, Y_1 \dots Y_n] \approx_B recY.D[Y, Y_1 \dots Y_n]$.

Let us define the relation $\mathcal{S} \subseteq \mathcal{C}_s \times \mathcal{C}_s$ as follows:

$$\begin{aligned} \mathcal{S} = \; & \{ (G[recY.C[Y]], G[recY.D[Y]]) \mid \\ & C, D, G \in \mathcal{C}_s,\ C \approx_B D, \\ & \text{and } G \text{ contains at most one variable} \}. \end{aligned}$$

Note that, since we assumed that C and D have at most the single free variable Y, the variables that occur bound in G do not occur free in C and D.

It will be enough to show that $\mathcal{S}$ is a weak bisimulation up to $\approx_B$. From this it follows $recY.C[Y] \approx_B recY.D[Y]$, by taking $G \equiv X$.

We prove that if $G[recY.C[Y]] \xrightarrow{a} P$ then there exist $Q, Q' \in \mathcal{C}_s$ with $(P, Q') \in \mathcal{S}$ and $G[recY.D[Y]] \overset{\hat{a}}{\Longrightarrow} Q \approx_B Q'$.

The converse follows by the symmetry of $\mathcal{S}$.

We prove the claim by induction on the depth d of the inference used to obtain $G[recY.C[Y]] \xrightarrow{a} P$.

Base: $d = 0$.
If $G[recY.C[Y]] \xrightarrow{a} P$ with an inference of depth 0, then the rule "Prefix" has been applied, and $G[X] \equiv a.G'[X]$, so $P \equiv G'[recY.C[Y]]$, with $G' \in \mathcal{C}_s$. Hence, also $G[recY.D[Y]] \equiv a.G'[recY.D[Y]] \xrightarrow{a} G'[recY.D[Y]]$ and we have that $(G'[recY.C[Y]], G'[recY.D[Y]]) \in \mathcal{S}$, as required.

Induction step. We proceed by cases on the structure of the context G:

- $G \in \mathcal{E}$. We have $G[recY.C[Y]] \equiv G[recY.D[Y]] \equiv G$, hence we immediately obtain the thesis.

- $G \equiv X$. Then $recY.C[Y] \xrightarrow{a} P$ has been deduced by applying the "Recursion" rule at the last step. So $C[recY.C[Y]] \xrightarrow{a} P$ with a shorter inference. Hence, by induction there exist $Q, Q' \in \mathcal{C}_s$ such that $C[recY.D[Y]] \overset{\hat{a}}{\Longrightarrow} Q \approx_B Q'$ with $(P, Q') \in \mathcal{S}$. But also $C[Y] \approx_B D[Y]$ and thus $D[recY.D[Y]] \overset{\hat{a}}{\Longrightarrow} Q'' \approx_B Q$. Since, $D[recY.D[Y]] \approx_B recY.D[Y]$, we have that it holds $recY.D[Y] \overset{\hat{a}}{\Longrightarrow} Q'''$ with $Q''' \approx_B Q'' \approx_B Q \approx_B Q'$.

- $G \equiv \sum_i a_i.G_i$. Then $\sum_i a_i.G_i[recY.C[Y]] \xrightarrow{a} P$ by applying the "Sum" in the last step. So, there exists i such that $a_i.G_i[recY.C[Y]] \xrightarrow{a} P$. Hence, it must be $P \equiv G_i[recY.C[Y]]$, with $G_i \in \mathcal{C}_s$. By applying the same rules, $G[recY.D[Y]] \overset{\hat{a}}{\Longrightarrow} Q \equiv G_i[recY.D[Y]]$, and $(P, Q) \in \mathcal{S}$.

- $G \equiv G_1 \setminus v$. Then $G_1[recY.C[Y]] \setminus \xrightarrow{a} P$ by applying the rule "Restriction" in the last step. So, $P \equiv P' \setminus v$, $a \notin v$ and $G_1[recY.C[Y]] \xrightarrow{a} P'$ by a shorter inference. By induction on $G_1 \in \mathcal{C}_s$, there exist $Q, Q' \in \mathcal{C}_s$ such that $G_1[recY.D[Y]] \overset{\hat{a}}{\Longrightarrow} Q \approx_B Q'$ with $(E', Q') \in \mathcal{S}$. Hence, we conclude $G_1[recY.D[Y]] \setminus v \overset{\hat{a}}{\Longrightarrow} Q \setminus v$, with $Q \setminus v \approx_B Q' \setminus v$ and $(P, Q' \setminus v) \in \mathcal{S}$ by construction of $\mathcal{S}$. In fact, $(P', Q') \in \mathcal{S}$ implies that there exists a context $H[X]$, with only a free variable X, such that $P' \equiv H[recY.C[Y]]$ and $Q' \equiv H[recY.D[Y]]$. Hence, $P \equiv P' \setminus v \equiv H[recY.C[Y]] \setminus v$ and $Q' \setminus v \equiv H[recY.D[Y]] \setminus v$.

- $G \equiv G_1[f]$. Then $G_1[recY.C[Y]][f] \xrightarrow{a} P$ by applying the rule "Relabelling" in the last step. So, $P \equiv P'[f]$, $a = f(a')$, and $G_1[recY.C[Y]] \xrightarrow{a'} P'$ by a shorter inference. By induction there exist $Q, Q' \in \mathcal{C}_s$ such that it holds $G_1[recY.D[Y]] \overset{\widehat{a'}}{\Longrightarrow} Q \approx_B Q'$ with $(P', Q') \in \mathcal{S}$ Hence, we conclude $G_1[recY.D[Y]][f] \overset{\widehat{f(a')}}{\Longrightarrow} Q[f]$, with $Q[f] \approx_B Q'[f]$ and $(P, Q'[f]) \in \mathcal{S}$ by construction.

- $G \equiv recZ.G_1[X, Z]$. Then we have that $recZ.G_1[recY.C[Y], Z] \xrightarrow{a} P$ by applying the rule "Recursion" in the last step. So, $G_1[recY.C[Y], recZ.G_1[recY.C[Y], Z]] \xrightarrow{a} P$ with a shorter inference. Hence, by induction there exist $Q, Q' \in \mathcal{C}_s$ such that $G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \overset{\hat{a}}{\Longrightarrow} Q \approx_B Q'$ with $(P, Q') \in \mathcal{S}$.
Since $G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \approx_B recZ.G_1[recY.D[Y], Z]$ we can conclude that $G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \overset{\hat{a}}{\Longrightarrow} Q'' \approx_B Q \approx_B Q'$. □

The next Lemma is used in the proof of Lemma 4.11.

Lemma A.1 Let $C \in \mathcal{C}_s$. Then $recY.(C \setminus H) \setminus H \approx_B (recY.C) \setminus H$.

PROOF. Without loss of generality we can assume that only the variable Y occurs free in the context C. The general case follows by Definition 2.5. Let $\mathcal{S}$ be defined as

$$\{(G[(recY.(C \backslash H))] \backslash H, G[recY.C] \backslash H) \mid G[X], C \in \mathcal{C}_s\}.$$

We prove that $\mathcal{S}$ is a strong bisimulation. From this the result follows by considering $G[X] \equiv X$.

Note that, since C has at most the single free variable Y, the variables that occur bound in G do not occur free in C.

To prove that $\mathcal{S}$ is a strong bisimulation we prove:

(1) if $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ then there exists Q such that $(P, Q) \in \mathcal{S}$, and $G[recY.C] \setminus H \xrightarrow{a} Q$

(2) if $G[recY.C] \setminus H \xrightarrow{a} Q$ then there exists P such that $(P, Q) \in \mathcal{S}$ and $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$

for any pair $(G[recY.(C \setminus H)] \setminus H, G[recY.C] \setminus H)$ in $\mathcal{S}$.

The proof proceeds by induction on the depth d of the inference used to prove $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ or $G[recY.C] \setminus H \xrightarrow{a} Q$.

Base: $d = 1$.
(1) If $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ with an inference of depth 1, then the rules "Restriction" and "Prefix" have been applied. Hence, we have $G[X] \equiv a.G'[X]$ and $P \equiv G'[recY.(C \setminus H)] \setminus H$. By applying the same rules to $G[recY.C] \setminus H$ we obtain that $G[recY.C] \setminus H \xrightarrow{a} Q$ with $Q \equiv G'[recY.C] \setminus H$. Since $G \in \mathcal{C}_s$, it holds that $G' \in \mathcal{C}_s$, hence $(P, Q) \in \mathcal{S}$.
(2) Similar to the previous case.

Induction step. We proceed by cases on the structure of the context $G[X]$.

- $G[X] \in \mathcal{E}$. Trivial.

- $G[X] \equiv X$.

(1) Let $(recY.(C \setminus H)) \setminus H \xrightarrow{a} P$. Then $P \equiv P' \setminus H$ and $C[recY.(C \setminus H)] \setminus H \xrightarrow{a} P'$ by a shorter inference. This implies that P' is free from high level action, i.e. $P \equiv P' \setminus H \equiv P'$. Hence, by inductive hypothesis, there exists Q such that $(P, Q) \in \mathcal{S}$, and $C[(recY.C)] \setminus H \xrightarrow{a} Q$. So, $(P, Q) \in \mathcal{S}$ and $(recY.C) \setminus H \xrightarrow{a} Q$.

(2) Let $(recY.C) \setminus H \xrightarrow{a} Q$. Then $Q \equiv Q' \setminus H$ and $C[recY.C] \setminus H \xrightarrow{a} Q$ by a shorter inference. Hence, by inductive hypothesis, there exists P such that $(P, Q) \in \mathcal{S}$ and $C[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$. So, $(P, Q) \in \mathcal{S}$ and $(recY.(C \setminus H)) \setminus H \xrightarrow{a} P$.

- $G[X] \equiv \sum_{i \in I} a_i.G_i[X]$.

(1) Let $G[recY.(C[Y] \setminus H)] \setminus H \xrightarrow{a} P$. Then $\exists i$ such that $a \equiv a_i$, $P \equiv G_i[recY.(C \setminus H)] \setminus H$. Hence, $G[recY.C] \setminus H \xrightarrow{a} Q$ with $Q \equiv G_i[recY.C] \setminus H$. From this, since $G_i[X] \in \mathcal{C}_s$, we immediately get $(P, Q) \in \mathcal{S}$.

(2) Let $G[recY.C]) \setminus H \xrightarrow{a} Q$. Then $\exists i$ such that $a \equiv a_i$, $Q \equiv G_i[recY.C] \setminus H$. Hence, $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ with $P \equiv G_i[recY.(C \setminus H)] \setminus H$. From this, since $G_i[X] \in \mathcal{C}_s$, we immediately get $(P, Q) \in \mathcal{S}$.

- $G[X] \equiv G_1[X] \setminus v$. Trivial.

- $G[X] \equiv G_1[X][f]$. Trivial.

- $G[X] \equiv recZ.G_1[X, Z]$.

(1) Let $G[recY.(C[Y] \setminus H)] \setminus H \xrightarrow{a} P$. In this case $recZ.G_1[recY.(C \setminus H), Z] \setminus H \xrightarrow{a} P$ and $G_1[recY.(C \setminus H), recZ.G_1[recY.(C \setminus H), Z]] \setminus H \xrightarrow{a} P$ by a shorter inference. By inductive hypothesis $G_1[recY.C, recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$ for some Q such that $(P, Q) \in \mathcal{S}$. Hence, we obtain $(P, Q) \in \mathcal{S}$ and $recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$, i.e., $G[recY.C] \setminus H \xrightarrow{a} Q$.

(2) Let $G[recY.C] \setminus H \xrightarrow{a} Q$ that is $recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$. Then, $G_1[recY.C, recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$, by a shorter inference. By inductive hypothesis $G_1[recY.(C \setminus H), recZ.G_1[recY.(C \setminus H), Z]] \setminus H \xrightarrow{a} P$ for some P such that $(P, Q) \in \mathcal{S}$. Therefore, $recZ.G_1[recY.(C \setminus H), Z] \setminus H \xrightarrow{a} P$. □

Proof of Lemma 4.11. Our hypothesis is that $C[E] \backslash H \approx_B C[E \backslash H] \backslash H$ and we have to prove that $(recY.C[E]) \backslash H \approx_B (recY.C[E \setminus H]) \setminus H$.

From the hypothesis and Lemma 4.10 we have that
$recY.(C[E] \setminus H) \approx_B recY.(C[E \setminus H] \setminus H)$.
By applying $\setminus H$ to both members we obtain
$recY.(C[E] \setminus H) \setminus H \approx_B recY.(C[E \setminus H] \setminus H) \setminus H$.
Notice that if $C[X] \in \mathcal{C}_s$, then also $C[E]$ and $C[E \setminus H]$ are in $\mathcal{C}_s$. Hence, we can apply Lemma A.1 to both members and get
$recY.(C[E]) \setminus H \approx_B recY.(C[E \setminus H]) \setminus H$,
i.e. our thesis. □

Proof of Lemma 4.15. Consider the binary relation $\mathcal{S} =$

$$\{((E|G) \setminus H, (F|K) \setminus H) \mid E, F, G, K \in P\_BNDC \text{ and } E \setminus H \approx_B F \setminus H, G \setminus H \approx_B K \setminus H\}.$$

It is easy to prove that $\mathcal{S}$ is a weak bisimulation. The only non-trivial case is the synchronization. Assume that $(E|G) \setminus H \xrightarrow{\tau} (E'|G') \setminus H$ with $E \xrightarrow{h} E'$ and $G \xrightarrow{\bar{h}} G'$. Then, since $E, G \in P\_BNDC$, we have $E \overset{\hat{\tau}}{\Rightarrow} E''$ with $E' \setminus H \approx_B E'' \setminus H$ and $G \overset{\hat{\tau}}{\Rightarrow} G''$ with $G' \setminus H \approx_B G'' \setminus H$. Hence, $E \setminus H \overset{\hat{\tau}}{\Rightarrow} E'' \setminus H$ and $G \setminus H \overset{\hat{\tau}}{\Rightarrow} G'' \setminus H$. By hypothesis we obtain $F \setminus H \overset{\hat{\tau}}{\Rightarrow} F' \setminus H$ with $F' \setminus H \approx_B E'' \setminus H$ and $K \setminus H \overset{\hat{\tau}}{\Rightarrow} K' \setminus H$ with $K' \setminus H \approx_B G'' \setminus H$. Hence, $(F|K) \setminus H \overset{\hat{\tau}}{\Rightarrow} (F'|K') \setminus H$ with $E', G', F', K' \in P\_BNDC$, $E' \setminus H \approx_B F' \setminus H$, and $G' \setminus H \approx_B K' \setminus H$, i.e. $((E'|G') \setminus H, (F'|K') \setminus H) \in \mathcal{S}$. □

Proof of Lemma 5.5. ($\Rightarrow$) If $E \in NDC$, then we have $(E|\Pi) \setminus H \approx_T E \setminus H$. Moreover, $E \setminus H$ is always in NDC and $E \setminus H \setminus H \approx_T E \setminus H$, hence $(E \setminus H|\Pi) \approx_T E \setminus H$. So by transitivity of $\approx_T$, we obtain that $(E|Pi) \setminus H \approx_T (E \setminus H|\Pi) \setminus H$.

($\Leftarrow$) Since $E \setminus H$ is always in NDC and $E \setminus H \setminus H \approx_T E \setminus H$, we obtain $(E|\Pi) \setminus H \approx_T (E \setminus H|\Pi) \setminus H \approx_T E \setminus H$. □

Proof of Theorem 5.7. Let E be a process in $\mathcal{P}$. From the fact that all the C_i are secure for $\mathcal{P}$ we obtain that for all $i \in I$ it holds $C_i[E] \setminus H \approx_T C_i[E \setminus H] \setminus H$. Hence, since $\approx_T$ is a congruence with respect to the non deterministic choice operator, $\sum_{i \in I}(C_i[E] \setminus H) \approx_T \sum_{i \in I}(C_i[E \setminus H] \setminus H)$. So, we can commute the restrictions with the sum and get $(\sum_{i \in I} C_i[E]) \setminus H \approx_T (\sum_{i \in I} C_i[E \setminus H]) \setminus H$. It trivially holds that $(\sum_{h_j \in H} h_j.D_j[E]) \setminus H \approx_T \mathbf{0} \approx_T (\sum_{h_j \in H} h_j.D_j[E \setminus H]) \setminus H$. Hence, again since $\approx_T$ is a congruence with respect to the non deterministic choice and the restriction operator commutes with the non deterministic choice we obtain $(\sum_{i \in I} C_i[E] + \sum_{h_j \in H} h_j.D_j[E]) \setminus H \approx_T (\sum_{i \in I} C_i[E \setminus H] + \sum_{h_j \in H} h_j.D_j[E \setminus H]) \setminus H$, i.e. our thesis. □

Proof of Lemma 5.8. We recall the following properties whose proofs are in [4]:

(1) if $E, G \in NDC$, then $E|G \in NDC$;

(2) $P \in NDC$ iff $P \setminus H \approx_T P/H$;

(3) $(E|G)/H \approx_T E/H|G/H$;

(4) if $E' \approx_T F'$ and $G' \approx_T K'$, then $E'|G' \approx_T F'|K'$.

Hence we obtain

$(E\|G) \setminus H$	$\approx_T$	by (1) and (2)
$(E\|G)/H$	$\approx_T$	by (3)
$(E/H\|G/H)$	$\approx_T$	by (2) and (4)
$(F/H\|K/H)$	$\approx_T$	by (3)
$(F\|K)/H$	$\approx_T$	by (1) and (2)
$(F\|K) \setminus H$.		

□