Secure Contexts for Information Flow Security

Annalisa Bossi, Damiano Macedonio, Carla Piazza, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari di Venezia
via Torino 155, 30172 Venezia, Italy
{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 levels. A usual requirement to ensure information flow security for a process is that no generic attacker can induce a high-to-low information flow. This requirement is too demanding when we have some knowledge about the contexts where the process is going to run. To deal with these situations we introduce the notion of *secure contexts* for a process. The notion is parametric with respect to both the observational equivalence and the operation used to characterize the low level behavior. In the paper 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, BNDC and NDC, are just special instances of our general notion.

1 Introduction

Information flow security in a multilevel system aims at guaranteeing that no high level information is revealed to users running at low levels [7, 8, 10], even in the presence of any possible malicious process. Our work starts from the observation that such a requirement could be too demanding when some knowledge about the context (environment) in which the process is going to run is available. In our approach the context can perform both high and low level action and can also incorporate possible attackers.

As an example, consider a Java applet E downloadable from the site of Money&Money ltd which should allow the Money&Money's customers (sellers) to get the price list of its products, while the rest of the world should only get the product list. The applet opens a window with two buttons: if one clicks on the first button, the product list is shown; otherwise, if one clicks on the second button, it is asked to insert a password and then the price list is shown. Let PWD_SELLER be the high level action representing the fact that E is waiting for a password from a customer before showing the price list through the high (or low) level output action $\overline{\text{PRICE\_LIST\_H}}$ (or $\overline{\text{PRICE\_LIST\_L}}$). If the password is not given, then E shows the product list through the high (or low) level output action $\overline{\text{PROD\_LIST\_H}}$ (or $\overline{\text{PROD\_LIST\_L}}$) [1]. E can be represented by a CCS-like

[1] Notice that when the product (or price) list is shown on the video both a low-level and a high-level user can read it. For this reason the output of the product (or price) list is represented by both a low and a high level action.

process of the form

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

Money&Money does not want the applet to be executed on a machine (context) C which reveals some high level information (the price list) to someone belonging to the rest of the world. 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

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

In this case high level information is revealed: when a low level user interacts with $C_1[E]$ he can read the price list, and thus it is reasonable to assume that C_1 is not 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 can read the price list, while low level ones can only read the product list:

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

In this case the flexibility of the context is obtained by splitting C_2 into two non-deterministic components: the first one manages the interactions with high level users and has in memory the seller'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 $C[E]$ by inserting the password, the price list becomes readable to low level observers. Does this really means that C is not secure for E? It depends on how strict we want to be. The high level user could have the permission to *downgrade* (see [11]) the level of the information stored in the price list.

The process E described here does not satisfy the basic information flow security properties such as *non-interference* [10] (also named *NDC* in [5]). However, it is reasonable to assume that the context C_2 is secure for E. To deal with these situations we introduce the notion of *secure context* for a process, which can be motivated both as security for the process and security for the process. This notion is parametric with respect to both an observational equivalence relation and an operation used to characterize the low level view of a process. Here, we consider weak bisimulation and trace equivalence. We show how to build secure contexts and prove that the security properties known as BNDC and NDC [5] are just special instances of our general security notion.

2 Basic Notions

The *Security Process Algebra* [5] is a variation of Milner's CCS [9], 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 to the variables occurring in a SPA term. More precisely, all the occurrences of the variable Z in $recZ.T$ are *bound*; and Z is *free* in a term T if there is an occurrence of Z in T which is not bound.

Definition 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 in $H \cup \{\tau\}$.*

A SPA term with free variables can be seen as an environment with places (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. *A SPA* context*, ranged over by $C, D, \ldots$, is a SPA term in which free variables can occur.*

We can consider a context also as a compound SPA constructor. In fact it can be used to build new 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: $E|E$.

We use the notation $C[Y_1, \ldots, Y_n]$ when we want to stress the fact that we are interested only in the free occurrences of the variables $Y_1, \ldots, Y_n$. The term $C[T_1, \ldots, T_n]$ is the term 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 $h.\mathbf{0}|(l.\bar{h}.\mathbf{0} + \tau.\mathbf{0})$, 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 $C[Y_1, \ldots, Y_n]$ does not imply neither that all the $Y_1, \ldots, Y_n$ occur free in the context nor that they include all the variables occurring free in the context.

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 inference rules depicted in Figure 1, where a is an action of Act, while ℓ belongs to $\mathcal{L}$.

Prefix $$\frac{-}{a.E \stackrel{a}{\rightarrow} E}$$

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

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

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

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

Recursion $$\frac{C[recZ.C[Z]] \stackrel{a}{\rightarrow} E'}{recZ.C[Z] \stackrel{a}{\rightarrow} E'}$$

Fig. 1. The operational rules for SPA

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 an internal action τ; $E \setminus v$ is a process E prevented from performing actions in v. $E[f]$ is the process E whose actions are renamed *via* the relabelling function f. Given a set v, the *hiding* operator mapping E into E/v, where the actions of $v \cup \bar{v}$ performed by E have been replaced by τ actions, can be defined using relabelling as follows: $E/v \equiv E[f_v]$, where $f_v(a) = a$, if $a \notin v \cup \bar{v}$ and $f_v(a) = \tau$, if $a \in v \cup \bar{v}$. Finally, $recZ.C[Z]$ can perform all the actions performed by the process obtained by substituting $recZ.C[Z]$ to the place-holder Z in the context $C[Z]$. Observe that in order to have $recZ.C[Z] \in \mathcal{E}$, Z is the only variable which can occur free in $C[Z]$.

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 $recZ.C[Z]$. This equivalence allows us to assume that whenever we substitute a context D to the free occurrences of X in $C[X]$, no free variable of D becomes bound in $C[D]$: an

α-conversion is always implicitly performed. For instance, by composing the contexts $C[Y] \equiv recX.Y$ and $D[X] \equiv a.X$, we obtain $C[D[X]] \equiv recZ.a.X$.

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 $\mathcal{E}$. The *weak bisimulation* relation [9] equates two processes if they are able to mutually simulate their behavior step by step. Weak bisimulation does not care about internal τ actions. The *trace equivalence* relation equates two processes if they have the same sets of traces, again, without considering the τ actions.

We will use the following auxiliary notations. If $t = a_1 \cdots a_n \in Act^*$ and $E \stackrel{a_1}{\rightarrow} \cdots \stackrel{a_n}{\rightarrow} E'$, then we write $E \stackrel{t}{\rightarrow} E'$. We also write $E \stackrel{t}{\Longrightarrow} E'$ if $E(\stackrel{\tau}{\rightarrow})^* \stackrel{a_1}{\rightarrow} (\stackrel{\tau}{\rightarrow})^* \cdots (\stackrel{\tau}{\rightarrow})^* \stackrel{a_n}{\rightarrow} (\stackrel{\tau}{\rightarrow})^* E'$ where $(\stackrel{\tau}{\rightarrow})^*$ 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, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\stackrel{\tau}{\rightarrow})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions).

Definition 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 $E \stackrel{a}{\rightarrow} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{R}$;
if $F \stackrel{a}{\rightarrow} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ 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 is an equivalence relation [9].

Definition 4 (Trace Equivalence). *For any process $E \in \mathcal{E}$ the set of traces $Tr(E)$ associated with E is defined as follows:*

$$Tr(E) = \{t \in \mathcal{L}^* \mid \exists E'\ E \stackrel{t}{\Longrightarrow} E'\}.$$

Two processes $E, F \in \mathcal{E}$ are trace equivalent*, denoted by $E \approx_T F$, if $Tr(E) = Tr(F)$.*

It is possible to prove that if two processes are weak bisimilar, then they are also trace equivalent, while the other implication does not hold.

Following [9] we extend the binary relations defined on processes to contexts as follows.

Definition 5 (Relations on Contexts). *Let $\mathcal{R}$ be a binary relation on $\mathcal{E}$. Let C and D be two contexts and assume $Y_1, \ldots, Y_n$ includes all their free variables. We say that $C\ \mathcal{R}\ D$, if for all set of processes $\{E_1, \ldots, E_n\}$ it holds*

$$C[E_1, \ldots, E_n]\ \mathcal{R}\ D[E_1, \ldots, E_n].$$

In the case of weak bisimulation, applying the above definition we have that two contexts are weak bisimilar if all the processes obtained by instantiating their variables are pair-wise 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 weak 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 5 requires that we instantiate 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 used to characterize the low level behavior, say E_l, of a process E (e.g., $E\backslash H$, E/H), and an observational 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 6 (Secure Contexts for a Class of Processes). *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*

$$\forall C[X] \in \mathcal{C}, \forall E \in \mathcal{P} \;\; C[E] \sim_l C[E_l].$$

In our definition the variable X is used to determine which are the "places" in C which have to be filled in using E. It can be that X does not occur free in C. In this case C is trivially secure (by reflexivity of $\sim_l$). Moreover, in C there can be other free variables different from X. In this case we have to apply Definition 5 and instantiate the other free variables in all the possible ways.

Example 1. 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 $C \in \mathcal{C}$ from the interactions between E_l and C. It is an instance of the *noninterference* schema proposed in [7]. In fact, no high level information can flow from E_l and the context C represents the environment in which E is executed, i.e. it *subsumes* a possible situation with actions performed by high and low level users as well as by possibly attackers.

Let us analyze the definition in the case in which only one process and one context are involved. The definition can be seen from two points of view: security

for the processes and security for the contexts. 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 user 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), since E is not able to reveal any high level information of C. In fact, E is able to reveal the same information which can be revealed by E_l and, since E_l cannot perform high level action, it cannot reveal any high information. We explain the two points of view with some examples.

Example 2 (Security for the Process). Consider the process representing a client of a bank using his card in a ATM (Automatic Teller Machine) to take money from his account. When the card is inserted in the ATM the code of the card is read and the client input his PIN code, then if the PIN is correct he can ask for the money. All the actions involved concern high level exchange of information between the client and the bank. We can formalize the process representing the client in front of the ATM as follows

$$\overline{\text{CARD\_CD}}.\overline{\text{PIN\_CD}}.\text{MONEY}.\mathbf{0},$$

where all the actions are high level actions. A *correct* ATM should read the codes, and if they are correct, give the money to the client. Hence, leaving out the details concerning the checks on the codes, it should be of the form

$$X|\text{CARD\_CD}.\text{PIN\_CD}.\overline{\text{MONEY}}.\mathbf{0}.$$

In this case, intuitively, the process is secure inside the context. In fact, since all the actions are high no information is revealed to the low level observer.

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. After one week the engineer removes the laptop and starts to make up counterfeit card. In this case the context in which the client inserts his card has been modified, i.e. a malicious component which reveals information to the low level observer (the engineer) has been added. The card and the PIN codes are first read by the malicious process, which both records and send them to the bank. The action of recording the codes is a low level action, since it can be used later by the low level observer to steal money. The codes are sent to the bank so that the client receives the money and does not suspect the fraud. The counterfeit context can be represented as follows

$$\begin{aligned} X|&\text{CARD\_CD}.\text{PIN\_CD}.\overline{\text{RECORD\_CDS}}.\overline{\text{CARD\_CD}}.\overline{\text{PIN\_CD}}.\mathbf{0}| \\ &\text{CARD\_CD}.\text{PIN\_CD}.\overline{\text{MONEY}}.\mathbf{0}. \end{aligned}$$

Obviously this context is not secure for the process. However, this does not mean that we give up using cards and ATMs. We just hope to use them in secure contexts.

Example 3 (Security for the Context). 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 buy a program E which is able to check on the stock market, using an Internet connection, read the files and perform some computations (using the information taken from the market) to determine whether the investments are profitable, and, if necessary (the investment are going bad), to check again on the stock market, for better opportunities. The second check on the stock market is recommended to suggest in the best possible way (using the last quotations), i.e. it is preferable not to use the cached stock market's quotations. Obviously Mr Earner does not want that someone knows if his investments are good or not. Since Mr Earner is not able to evaluate the quality of his investments he only knows that his machine is in one of the following two situations:

$$X \mid \overline{\text{ACCESS\_GOOD}}.\mathbf{0} \qquad \text{or} \qquad X \mid \overline{\text{ACCESS\_BAD}}.\text{SUGGESTIONS}.\mathbf{0}.$$

In the first case Mr Earner investments are good and this fact can be revealed through the high level output $\overline{\text{ACCESS\_GOOD}}$. In the second case Mr Earner investments are bad, hence after the high level output its machine is ready to have in input some suggestions through the high level input SUGGESTIONS. Mr Earner wants E to be secure w.r.t. both contexts. Let us assume that Mr Earner investments are good, i.e. we consider the first context[2].

If the program E is of the form

$$\begin{aligned}&\text{CHECK\_MARKET}.(\text{ACCESS\_GOOD}.\mathbf{0} + \\ &\qquad \text{ACCESS\_BAD}.\text{CHECK\_MARKET}.\overline{\text{SUGGESTIONS}}.\mathbf{0}),\end{aligned}$$

where the only low level action is the input CHECK_MARKET, then, by observing that E has not checked a second time on the stock marked, someone could be able to deduce that Mr Earner's investments are good. Hence, E is not secure.

The program

$$\begin{aligned}&\text{CHECK\_MARKET}.(\text{ACCESS\_GOOD}.\text{CHECK\_MARKET}.\mathbf{0} + \\ &\qquad \text{ACCESS\_BAD}.\text{CHECK\_MARKET}.\overline{\text{SUGGESTIONS}}.\mathbf{0})\end{aligned}$$

is secure, because, by performing the check also when the investments are good, it does not reveal anything about them.

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

$$\text{CHECK\_MARKET}.(\text{ACCESS\_GOOD}.\mathbf{0} + \text{ACCESS\_BAD}.\overline{\text{SUGGESTIONS}}.\mathbf{0}).$$

It performs the low level input only once before analyzing the situation of the investments (i.e., it suggests using the cached dates). Hence, it is secure.

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

Notice, that this example recalls the case of military radio transmissions. In order to avoid that someone knows when some information have been transmitted, every n instants a message is sent. Only one of the messages contains the real information.

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.

Proposition 1. *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 X-secure for $\mathcal{P}_1$ with respect to X.*

Definition 6 introduces a general security notion. To analyze it more concretely it is necessary to instantiate the observational equivalence $\sim_l$ and the operation defining E_l. In order to get instances useful in the practical cases, a decidable equivalence and a computable operation are the minimal reasonable requirements. However, they are not strong enough to guarantee the decidability of the security notion. In fact, Definition 6 involves two universal quantifications which imply that if either $\mathcal{C}$ or $\mathcal{P}$ are infinite the definition is not operative.

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

First we analyze the properties of our security definition by instantiating the observational equivalence ($\sim$) and the low level view of a process (E_l) in the following way. Let $E, F \in \mathcal{E}$

- $E \sim F$ iff $E \approx_B F$;
- E_l is $E \setminus H$.

Notice that in this way $\sim_l$ is an equivalence relation.

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 iff

$$\forall C[X] \in \mathcal{C}, \forall E \in \mathcal{P}\; C[E] \setminus H \approx_B C[E \setminus H] \setminus H.$$

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

Example 4. Let us consider again the process and the contexts in the Introduction. Since, E has the form

$$\begin{aligned} \text{PWD\_SELLER}.(\overline{\text{PRICE\_LIST\_H}}.\mathbf{0} + \overline{\text{PRICE\_LIST\_L}}.\mathbf{0}) + \\ (\overline{\text{PROD\_LIST\_H}}.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}). \end{aligned}$$

where PWD_SELLER, $\overline{\text{PRICE\_LIST\_H}}$, and $\overline{\text{PROD\_LIST\_H}}$ are high level actions, we obtain that $E \setminus H$ is

$$\overline{\text{PROD\_LIST\_L}}.\mathbf{0}.$$

The first context C we considered is

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

Hence, we obtain that $C[E] \setminus H$ is

$$\tau.\overline{\text{PRICE\_LIST\_L}}.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$$

This is not weak bisimilar to $C[E \setminus H] \setminus H$ which is

$$\overline{\text{PROD\_LIST\_L}}.\mathbf{0}$$

Hence, the context is not secure. Indeed, a low level user interacting with $C[E]$ can obtain confidential information.

The second context we considered is

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

In this case we have that both $C[E] \setminus H$ and $C[E \setminus H] \setminus H$ are bisimilar to

$$\text{PWD\_LOW}.\overline{\text{PROD\_LIST\_L}}.\mathbf{0}$$

The context is secure. Indeed, only a further interaction of a high level user can allow the low level observer to read the price list (downgrading of information).

Example 5. In Example 3 we said that the second program E is secure in both the proposed contexts. In fact, E never reveals to a low level user the situation of Mr Earner's investments, since a second check on the marked is performed in any case. However, with this instance of our framework and using the first context of Example 3 we obtain that $C[E] \setminus H$ is

$$\text{CHECK\_MARKET}.(\tau.\text{CHECK\_MARKET}.\mathbf{0}),$$

while $C[E \setminus H] \setminus H$ is

$$\text{CHECK\_MARKET}.\mathbf{0},$$

hence the security property does not hold. This models the following situation: the low level user can deduce that the quality of the investments is still under evaluation, by observing that the second check on the stock marked has not yet been performed. Nevertheless, if we assume that the evaluation always takes a constant amount of time, the process can be safely executed inside the context, since from the fact that the second check on the market is performed the low level observer cannot deduce which high level synchronization is occurred (the good or the bad one). If we consider the instance of our security property obtained by using the hiding operator instead of the restriction one (see [4]), i.e., we require

$$C[E]/H \approx_B C[E/H]/H$$

we obtain that $C[E]/H$ and $C[E/H]/H$ are both weak bisimilar to

$$\text{CHECK\_MARKET.CHECK\_MARKET.}\mathbf{0},$$

hence the security property holds.

The third program of Example 3 satisfies $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$, w.r.t. 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 notion known as *BNDC*. The *BNDC* [4] security property aims at guaranteeing that no information flow from the high to the low level is possible, even in the presence of any malicious high level process. 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. 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 system 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 the system E.

Definition 7 (BNDC). *Let $E \in \mathcal{E}$.*

$$E \in BNDC \;\; \textit{iff} \;\; \forall\, \Pi \in \mathcal{E}_H,\; E \setminus H \approx_B (E|\Pi) \setminus H.$$

Example 6. The *BNDC* property is powerful enough to detect information flows due to the possibility for a high level malicious process to block or unblock a system. Let $H = \{h\}$, $L = \{l, j\}$ and $E_1 l.h.j.\mathbf{0} + l.j.\mathbf{0}$. Consider the process $\Pi \equiv \bar{h}.\mathbf{0}$. We have that $(E_1|\Pi) \setminus H \approx_B l.j.\mathbf{0}$, while $E_1 \setminus H \approx_B l.\mathbf{0} + l.j.\mathbf{0}$. Note that the latter may (nondeterministically) block after the l input. Having many instances of this process, a low level user could deduce if $\bar{h}$ is executed by observing whether the system always performs j or not. Process E_1 may be "repaired", by including the possibility of choosing to execute j or not inside the process. Indeed, process $E_2 \equiv l.h.j.\mathbf{0} + l.(\tau.j.\mathbf{0} + \tau.\mathbf{0})$ is *BNDC*.

The following lemma states that all the contexts of the form $X|\Pi$ with $\Pi \in \mathcal{E}_H$ are secure for *BNDC* processes.

Lemma 1. *Let $E \in \mathcal{E}$.*

$$E \in BNDC \textit{ iff } C[E] \setminus H \approx_B C[E \setminus H] \setminus H$$

for all contexts $C[X] \equiv X|\Pi$ with $\Pi \in \mathcal{E}_H$.

Proof. See Appendix. □

Notice that a *BNDC* process can be safely executed also in a context in which an external attacher is able to guess all the high level password. When strict policies are applied on choices and changes of the passwords, the requirement of *BNDC* could be too demanding. In this sense it becomes interesting to study also processes which are secure only in a more restricted class of contexts.'

Example 7. The process in the introduction (see also Example 4) is not a *BNDC* process. In fact, the context $X|\overline{\text{PWD\_SELLER}}.\mathbf{0}$ is a context of the form $X|\Pi$ with $\Pi \in \mathcal{E}_H$ and it is not secure for E, hence by Lemma 1 we obtain that E is not *BNDC*. However, as shown in Example 4, there are complex contexts in which E can be safely executed.

The second process of Example 3 is not a *BNDC* process. In fact, the context $X|\overline{\text{ACCESS\_GOOD}}.\mathbf{0}$ is not secure for it. More in general, we can observe that if $l \in L$, then a process of the form

$$\sum_{h \in H} h.l.\mathbf{0}$$

is not *BNDC*. In this case the information which flow to the low level observer is only that a high level action has (or has not) been performed. However, when the low level user is able to perform his action he is not able to infer which one of the high level action has been chosen.

The third process of Example 3 can be proved to be *BNDC*.

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 they admit larger classes of secure contexts.

4.1 Secure Contexts for a generic class $\mathcal{P}$

Our first result can be immediately proved by applying the definitions.

Theorem 1. *Let $\mathcal{P}$ be a class of processes. The following contexts are secure for $\mathcal{P}$ with respect to X.*

- *$F \in \mathcal{E}$;*
- *Y, with Y a variable*[3]*;*
- *$\sum_{l_i \in L} l_i.C_i + \sum_{h_j \in H} h_j.D_j$, with the C_i's secure for $\mathcal{P}$ with respect to X;*
- *$C \setminus v$, with C secure for $\mathcal{P}$ with respect to X;*
- *$C[f]$, with C 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 $\mathcal{P}$.

Example 8. Consider the process $E \equiv h.l.\mathbf{0} + \bar{h}.\mathbf{0}$. Let the class $\mathcal{P} = \{E\}$. The context X is secure for $\mathcal{P}$ (see Theorem 1), but the context $X|X$ is not secure for $\mathcal{P}$.

[3] Y can also be the variable X.

Notice that Theorem 1 is not 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 Theorem 1 does not tell us how to prove that C is secure for $\mathcal{P}$.

Now 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 closed (see Example 8).

Definition 8 (The Class $\mathcal{C}_s$). *Let $\mathcal{C}_s$ be the minimum class such that:*

- *if $F \in \mathcal{E}$, then $F \in \mathcal{C}_s$;*
- *if Y is a variable, then $Y \in \mathcal{C}_s$;*
- *if $C_i \in \mathcal{C}_s$ for all $i \in I$, then $\sum_{i \in I} a_i.C_i \in \mathcal{C}_s$;*
- *if $C \in \mathcal{C}_s$, then $C \setminus v \in \mathcal{C}_s$;*
- *if $C \in \mathcal{C}_s$, then $C[f] \in \mathcal{C}_s$;*
- *if $C \in \mathcal{C}_s$, then $recY.C \in \mathcal{C}_s$.*

Notice that we impose to the sums to be guarded in the contexts, but not in the processes.

It is easy to define a proof system whose proofs correspond exactly to the constructions of the contexts of $\mathcal{C}_s$.

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

Example 9. The context $a.X + b.Y + c.Z$ is in $\mathcal{C}_s$. Hence, also the context $recY.(recZ.(a.X + b.Y + c.Z))$ belongs to $\mathcal{C}_s$.

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

Theorem 2. *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. See Appendix. □

Example 10. 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. Similarly, all the low level actions are defined. Hence, C has the form

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

Since C belongs to $\mathcal{C}_s$, the program coming from the web is secure inside C.

As shown by Example 8, 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 the process and the context. For this reason in the next subsection we concentrate on classes of processes on which we prove that some contexts involving the parallel operator are secure.

4.2 Secure Contexts for sub-classes of *BNDC*

As stated in Lemma 1 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 [3, 6, 2]). In particular, in [2] three of these sufficient conditions have been considered and it has been shown that they can be parametrically characterized with respect to an opportune bisimulation relation. In virtue of Proposition 1 all the contexts which are secure for the larger of these three classes, i.e. *P_BNDC*, are secure also for the other two classes. *P_BNDC* is nothing but the persistent version of *BNDC*. The persistency of *P_BNDC* has been proved to be fundamental to deal with dynamic contexts (see [6]).

Definition 9 (P_BNDC). *Let $E \in \mathcal{E}$.*

$$E \in P\_BNDC \;\; \textit{iff} \;\; \forall E' \textit{ reachable from } E \; E' \in BNDC.$$

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

Lemma 2. *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 us that by requiring to a context to map *P_BNDC* processes into *P_BNDC* processes we obtain that the parallel composition of secure contexts is secure. More in general we can introduce the following definition, which will turn out to be useful also in the next section.

Definition 10 ($\mathcal{P}$-contexts). *Let $\mathcal{P}$ be a class of processes and $C[X, \bar{Y}]$ be a context whose free variables are in $X \cup \bar{Y}$. $C[X, \bar{Y}]$ is said to be a* $\mathcal{P}$-context with respect to *X if for all $E \in \mathcal{P}$ and for all $\bar{F} \in \mathcal{E}$ it holds that $C[E, \bar{F}] \in \mathcal{P}$.*

Definition 11. *A context $C[X]$ is said to be* *P_BNDC*-secure with respect to *X if it is a P_BNDC-context with respect to X and it is secure for P_BNDC with respect to X.*

Theorem 3. *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. See Appendix. □

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

From Proposition 1 we have that the contexts which can be proved to be secure using Theorem 3 are secure also for *SBNDC* (see [3]) and *CP_BNDC* (see [2]) processes. In fact, in [2] it has been proved that these two are subclasses of *P_BNDC*.

Example 11. Consider the third program of Example 3

$$\text{CHECK\_MARKET}.(\text{ACCESS\_GOOD}.\mathbf{0} + \text{ACCESS\_BAD}.\overline{\text{SUGGESTIONS}}.\mathbf{0}).$$

This process is *P_BNDC*, hence by applying Theorem 3 we immediately get that the two contexts of Example 3 are secure for the process.

Example 12. Let $a \in \mathcal{L}$ be an action and E be a *P_BNDC* process in which neither a nor $\bar{a}$ occur. Let $\mathcal{P}$ be a class of *P_BNDC* processes whose termination is announced by the execution of an END action. Consider the context C defined as

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

When in C we replace the variable X with a process F taken from $\mathcal{P}$ we obtain that F is executed and then E is executed, i.e. we have obtained a context which behaves like a sequential operator. From Theorem 3 we have that $X|\bar{a}.E$ is secure for $\mathcal{P}$. Hence, from Theorem 1, we obtain that C is secure for $\mathcal{P}$.

However, Theorem 3 is not 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*. A decidable class of contexts which are *P_BNDC*-contexts is characterized by the following definition.

Definition 12 (The Class $\mathcal{C}_p$). *Let $\mathcal{C}_p$ be the minimum class such that:*

- *if $F \in$ P_BNDC, then $F \in \mathcal{C}_p$;*
- *the variable X is in $\mathcal{C}_p$;*
- *if Y is a variable, then $Y \setminus H$ and Y/H are in $\mathcal{C}_p$;*
- *if $C_i, D_j \in \mathcal{C}_p$, $i \in I$ and $j \in J$, then $\sum_{i\in I} l_i.C_i + \sum_{j\in J}(h_j.D_j + \tau.D_j) \in \mathcal{C}_p$, where $l_i \in L$ and $h_j \in H$;*
- *if $C, D \in \mathcal{C}_p$ then $C|D \in \mathcal{C}_p$*
- *if $C \in \mathcal{C}_p$, then $C \setminus v \in \mathcal{C}_p$;*
- *if $C \in \mathcal{C}_p$, then $C[f] \in \mathcal{C}_p$;*

Theorem 4. *If $C[X] \in \mathcal{C}_p$ then $C[X]$ is P_BNDC-secure with respect to X.*

Proof. See Appendix. □

The class $\mathcal{C}_p$, in a certain sense, corresponds to the class of processes defined by the proof system *Core* described in [1]. In fact, the high level prefixes are *controlled* by the τ ones, like in *Core*. Moreover, it is not possible to use the recursion operator. It could be interesting to study if the same extensions to the cases with recursion presented in [1] can be added here.

5 Second Instance: Trace Equivalence and Restriction

Sometimes weak bisimulation is considered to be too demanding, i.e. in some cases processes which are not weak bisimilar can be considered equivalent.

Example 13. The process in the introduction was of the form

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

Money&Money 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

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

If we consider the first context

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

we obtain that $C[E] \setminus H$ is

$$\tau.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$$

This is not weak bisimilar to $C[E \setminus H] \setminus H$ which is

$$\overline{\text{PROD\_LIST\_L}}.\mathbf{0}$$

However, the low level user cannot read the price list using this context. The information which flow to the low level user is that if he cannot read the product list, then a high level user has used the applet to read the price list. But, in this case the applet has terminated and it has disappeared. It really seems that in this case the use of bisimulation is too restrictive, while trace-equivalence could be the right choice.

In this section we consider the following instance:

- $E \sim F$ iff $E \approx_T F$;
- E_l is $E \setminus H$.

In this case a class of contexts $\mathcal{C}$ is secure for a class of processes $\mathcal{P}$ with respect to X iff

$$\forall C[X] \in \mathcal{C}, \forall E \in \mathcal{P}\ C[E] \setminus H \approx_T C[E \setminus H] \setminus H.$$

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

Example 14. Using the above definition the context of Example 13 is secure.

Let us consider the security notion known as *NDC* which is defined as *BNDC*, but using trace equivalence instead of weak bisimulation.

Definition 13 (NDC). *Let* $E \in \mathcal{E}$.

$$E \in NDC \;\; \textit{iff} \;\; \forall\, \Pi \in \mathcal{E}_H,\; E \setminus H \approx_T (E|\Pi) \setminus H.$$

The *NDC* security property is decidable as it immediately follows from the following characterization, whose proof can be found in [3].

Lemma 3. *Let* $E \in \mathcal{E}$.

$$E \in NDC \quad \textit{iff} \quad E/H \approx_T E \setminus H.$$

Example 15. The process in the introduction is not *NDC*.

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* contexts.

Lemma 4. *Let* $E \in \mathcal{E}$.

$$E \in NDC \textit{ iff } C[E] \setminus H \approx_T C[E \setminus H] \setminus H$$

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 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. *Let* $\mathcal{C}$ *be a class and* $\mathcal{P}$ *be a class of processes. If* $\forall C[X] \in \mathcal{C}, \forall E \in \mathcal{P}\; C[E]\setminus H \approx_B C[E\setminus H]\setminus H$, *then* $\forall C[X] \in \mathcal{C}, \forall E \in \mathcal{P}\; C[E]\setminus H \approx_T C[E\setminus H]\setminus H$.

Proof. See Appendix. □

This means that the class of contexts of Theorem 1 and the class of contexts $\mathcal{C}_s$ are secure for a generic class $\mathcal{P}$ of processes.

Moreover, it is easy to prove that the sum of secure contexts is secure.

Theorem 6. *Let* $\mathcal{P}$ *be a class of processes and* X *be a variable.* $\sum_{i\in I} C_i + \sum_{h_j \in H} h_j.D_j$ *is secure for* $\mathcal{P}$ *with respect to* X, *if for all* $i \in I$ C_i *is secure for* $\mathcal{P}$ *with respect to* X.

Proof. See Appendix. □

Notice that, again, 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 8 witnesses this fact.

5.2 Secure Contexts for NDC processes

Here we rediscover the equivalent 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 2.

Lemma 5. *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 14. *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 7. *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. See Appendix. □

Again Theorem 7 does not provide us a decidability procedure, because the definition of NDC-secure contexts is not operative. In the following definition we characterize a decidable class of NDC-contexts, which is the analogous of the class $\mathcal{C}_p$ of Definition 12.

Definition 15 (The Class $\mathcal{C}_n$). *Let $\mathcal{C}_n$ be the minimum class such that:*

- *if $F \in NDC$, then $F \in \mathcal{C}_n$;*
- *the variable X is in $\mathcal{C}_n$;*
- *if Y is a variable, then $Y \setminus H$ and Y/H are in $\mathcal{C}_n$;*
- *if $C \in \mathcal{C}_n$ and $l \in L$, then $l.C \in \mathcal{C}_n$;*
- *if $C \in \mathcal{C}_n$ and $h \in H$, then $h.C + \tau.C \in \mathcal{C}_n$;*
- *if $C, D \in \mathcal{C}_n$, then $C + D \in \mathcal{C}_n$;*
- *if $C, D \in \mathcal{C}_n$ then $C|D \in \mathcal{C}_n$*
- *if $C \in \mathcal{C}_n$, then $C \setminus v \in \mathcal{C}_n$;*
- *if $C \in \mathcal{C}_n$, then $C[f] \in \mathcal{C}_n$;*

Theorem 8. *If $C[X] \in \mathcal{C}_n$ then $C[X]$ is NDC-secure with respect to X.*

Proof. See Appendix. □

6 Conclusions

We presented a security notion for processes which is more flexible than some already known security properties where the attackers are all the possible contexts of the form $X|\Pi$, with $\Pi \in \mathcal{E}_H$. The flexibility is a consequence of the fact that our notion is parametric with respect to a class of contexts. On the one hand our notion can be used to restrict the possible attackers: e.g., when it is not reasonable to assume that all high level passwords can be guessed. On the other hand our notion allows to enlarge the set of possible attackers, since also low level actions can be performed and SPA operators can be freely combined in the context construction.

An interesting future issue could be the reformulation of our security property in richer languages (e.g., π-calculus). Such a reformulation would allow a more deep comparison with other approaches to security.

References

1. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. A Proof System for Information Flow Security. In M. Leuschel, editor, *Proc. of Int. Workshop on Logic Based Program Development and Transformation*, LNCS. Springer-Verlag, 2002. To appear.
2. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Bisimulation and unwinding for verifying possibilistic security properties. In *Int. Workshop on Verification Model Checking and Abstract Interpretation (VMCAI'03)*, LNCS. Springer-Verlag, 2003. To appear.
3. R. Focardi. *Analysis and Automatic Detection of Information Flows in Systems and Networks.* Ublcs-99-16, University of Bologna, 1999.
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*. Springer-Verlag, 2001.
6. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the IEEE Computer Security Foundations Workshop*, pages 307–319. IEEE Computer Society Press, 2002.
7. 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.
8. 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.
9. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
10. 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.
11. A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J. Selected Areas in Communication*, 21(1), 2003. To appear.

7 Appendix

Proof of Lemma 1

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

Proof of Theorem 1

- Since $F \setminus H \approx_B F \setminus H$ for each $F \in \mathcal{E}$, F is secure for $\mathcal{P}$.
- The fact that a variable Y is secure for $\mathcal{P}$ follows again from $E \setminus H \approx_B E \setminus H \setminus H$ for all $E \in \mathcal{E}$.
- 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, $C[E \setminus H] \setminus H \stackrel{\hat{b}}{\Rightarrow} C'''$ with $C'' \approx_B C'''$. So we obtain that $C[E \setminus H][f] \setminus H \stackrel{\hat{a}}{\Rightarrow} C'''[f]$ with $C''[f] \approx_B C'''[f]$.

□

The following lemma is useful to prove Lemma 8.

Lemma 6. *Let $C, D, C', D' \in \mathcal{C}_s$. If $C \approx_B D$ and $C' \approx_B D'$, then*

- $a.C + a'.C' \approx_B a.D + a'.D'$;
- $C \setminus v \approx_B D \setminus v$;
- $C[f] \approx_B D[f]$;
- $recY.C \approx_B recY.D$.

Proof. Given $C, D, C', D' \in \mathcal{C}_s$ with $C \approx_B D$, $C' \approx_B D'$, and $a, a' \in Act$, we can easily see that $a.C + a'.C' \approx_B a.D + a'.D'$. In fact, the only actions that the two processes can perform are a and a' Hence, they can only reduce themselves to $C \approx_B D$ or $C' \approx_B D'$.

The cases of "Restriction" and "Relabelling" are similar.

We prove the "Recursion" case. 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

$C[X]$ and $D[X]$ have at most the single free variable X. The general case follows from Definition 5.

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

$$\mathcal{S} = \{(G[recX.C[X]], G[recX.D[X]]) \mid C, D, G \in \mathcal{C}_s,\ C \approx_B D, \\ \text{and } G \text{ contains at most one variable}\}.$$

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

$$\text{If } G[recX.C[X]] \xrightarrow{a} P \text{ then there exist } Q, Q' \in \mathcal{C}_s \\ G[recX.D[X]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q', \text{ with } (P, Q') \in \mathcal{S}.$$

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 $G[recX.C[X]] \xrightarrow{a} P$.

Base: $d = 0$.

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

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

- $G \in \mathcal{E}$. We have $G[recX.C[X]] \equiv G[recX.D[X]] \equiv G$, hence we immediately obtain the thesis.
- $G \equiv Y$. Then $recX.C[X] \xrightarrow{a} P$ has been deduced by applying the "Recursion" rule at the last step. So $C[recX.C[X]] \xrightarrow{a} P$ with a shorter inference. Hence, by induction there exist $Q, Q' \in \mathcal{C}_s$ such that $C[recX.D[X]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q'$ with $(P, Q') \in \mathcal{S}$. But also $C[X] \approx_B D[X]$ and thus $D[recX.D[X]] \stackrel{\widehat{a}}{\Longrightarrow} Q'' \approx_B Q$. Since, $D[recX.D[X]] \approx_B recX.D[X]$, we have that it holds $recX.D[X] \stackrel{\widehat{a}}{\Longrightarrow} Q'''$ with $Q''' \approx_B Q'' \approx_B Q \approx_B Q'$.
- $G \equiv \sum_i a_i.G_i$. Then $\sum_i a_i.G_i[recX.C[X]] \xrightarrow{a} P$ by applying the "Sum" in the last step. So, there exists i such that $a_i.G_i[recX.C[X]] \xrightarrow{a} P$. Hence, it must be $P \equiv G_i[recX.C[X]]$, with $G_i \in \mathcal{C}_s$. By applying the same rules, $G[recX.D[X]] \stackrel{\widehat{a}}{\Longrightarrow} Q \equiv G_i[recX.D[X]]$, and $(P, Q) \in \mathcal{S}$.
- $G \equiv G_1 \setminus v$. Then $G_1[recX.C[X]] \setminus \xrightarrow{a} P$ by applying the rule "Restriction" in the last step. So, $P \equiv P' \setminus v$, $a \notin v$ and $G_1[recX.C[X]] \xrightarrow{a} P'$ by a shorter inference. By induction on $G_1 \in \mathcal{C}_s$, there exist $Q, Q' \in \mathcal{C}_s$ such that $G_1[recX.D[X]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q'$ with $(E', Q') \in \mathcal{S}$. Hence, we conclude $G_1[recX.D[X]] \setminus v \stackrel{\widehat{a}}{\Longrightarrow} Q \setminus v$, 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 $H[Z]$, with only a free variable Z, such that $P' \equiv H[recX.C[X]]$ and $Q' \equiv H[recX.D[X]]$. Hence, $P \equiv P' \setminus v \equiv H[recX.C[X]] \setminus v$ and $Q' \setminus v \equiv H[recX.D[X]] \setminus v$.

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

□

The following lemmas are at the basis of the proof of Theorem 2.

Lemma 7. *Let $C \in \mathcal{C}_s$.*

$$recY.(C \setminus H) \setminus H \approx_B (recY.C) \setminus H.$$

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 5.

Let $\mathcal{S} = \{(G[(recY.(C \setminus H))] \setminus H, G[recY.C] \setminus H) \mid G[X], C \in \mathcal{C}_s\}$. We prove that $\mathcal{S}$ is a strong bisimulation. ¿From this the result follows by considering $G[X] \equiv X$. To prove that $\mathcal{S}$ is a strong bisimulation we prove that for any pair $(G[recY.(C \setminus H)] \setminus H, G[recY.C] \setminus H)$ in $\mathcal{S}$

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

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

Base: $d = 1$.

1. If $G[recY.(C \setminus H)] \setminus H \stackrel{a}{\to} P$ with an inference of depth 1, then the rules "Restriction" and "Prefix" have been applied. Hence, $G[X] \equiv a.G'[X]$ and $P \equiv G'[recY.(C \setminus H)] \setminus H$. By applying the same rules to $G[recY.C] \setminus H$ we obtain that $G[recY.C] \setminus H \stackrel{a}{\to} Q$ 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. 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 $G[recY.(C[Y] \setminus H)] \setminus H \xrightarrow{a} P$. Then $recZ.G_1[recY.(C \setminus H), Z] \setminus H \xrightarrow{a} P$ 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 there exists Q such that $(P, Q) \in \mathcal{S}$, and $G_1[recY.C, recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$. Hence, $(P, Q) \in \mathcal{S}$ and $recZ.G_1[recY.C, Z]] \setminus H \xrightarrow{a} Q$, that is $G[recY.C] \setminus H \xrightarrow{a} Q$.
  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 there exists P such that $(P, Q) \in \mathcal{S}$ and $G_1[recY.(C \setminus H), recZ.G_1[recY.(C \setminus H), Z]] \setminus H \xrightarrow{a} P$. Hence, $recZ.G_1[recY.(C \setminus H), Z] \setminus H \xrightarrow{a} P$.

□

Lemma 8. *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. Our hypothesis is that $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ and we have to prove that $(recY.C[E]) \setminus H \approx_B (recY.C[E \setminus H]) \setminus H$.
From the hypothesis and Lemma 6 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 7 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 Theorem 2
By induction on the structure of C.

- $C \in \mathcal{E}$. We have already prove in Theorem 1, that C is secure for $\mathcal{P}$.
- $C \equiv Y$. Again, this has been proved in Theorem 1.
- $C \equiv \sum_{i \in I} a_i.C_i$. By induction on the C_i's and by Lemma 6 we have the thesis.
- $C \equiv C_1 \setminus v$. By induction and applying Lemma 6 we obtain the thesis.
- $C \equiv C_1[f]$. Again, by induction and Lemma 6 we get the thesis.
- $C \equiv recY.C_1$. By induction and Lemma 8 we have the thesis.

□

Proof of Lemma 2
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 $E \stackrel{\hat{\tau}}{\Rightarrow} E''$ with $E' \setminus H \approx_B E'' \setminus H$ and $G \stackrel{\hat{\tau}}{\Rightarrow} G''$ with $G' \setminus H \approx_B G'' \setminus H$. Hence, $E \setminus H \stackrel{\hat{\tau}}{\Rightarrow} E'' \setminus H$ and $G \setminus H \stackrel{\hat{\tau}}{\Rightarrow} G'' \setminus H$. By hypothesis we obtain $F \setminus H \stackrel{\hat{\tau}}{\Rightarrow} F' \setminus H$ with $F' \setminus H \approx_B E'' \setminus H$ and $K \setminus H \stackrel{\hat{\tau}}{\Rightarrow} K' \setminus H$ with $K' \setminus H \approx_B G'' \setminus H$. Hence, $(F|K) \setminus H \stackrel{\hat{\tau}}{\Rightarrow} (F'|K') \setminus H$ 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 Theorem 3
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 2 to these four processes we get the thesis. □

Proof of Theorem 4
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 [1], 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 [6]).

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 context. The basic steps are trivial. All inductive steps follow by Theorem 1 except parallel case, which follows from Lemma 2.

□

Proof of Lemma 4

($\Rightarrow$) If $E \in NDC$, then $(E|\Pi) \setminus H$ $approx_T E \setminus H$. 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

This is an immediate consequence of the fact that if $E \approx_B F$, then $E \approx_T F$, for all $E, F \in \mathcal{E}$. □

Proof of Theorem 6

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, we have that $\sum_{i \in I}(C_i[E] \setminus H) \approx_T \sum_{i \in I}(C_i[E \setminus H] \setminus H)$. 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

We recall the following properties whose proofs can be found in [3]:

(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

$$\begin{array}{lll} (E|G) \setminus H & \approx_T & \text{by (1) and (2)} \\ (E|G)/H & \approx_T & \text{by (3)} \\ (E/H|G/H) & \approx_T & \text{by (2) and (4)} \\ (F/H|K/H) & \approx_T & \text{by (3)} \\ (F|K)/H & \approx_T & \text{by (1) and (2)} \\ (F|K) \setminus H. & & \end{array}$$

□

Proof of Theorem 7

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 to these four processes we get the thesis. □

Proof of Theorem 8

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 1 except cases of parallel and nondeterministic choice. The parallel step follows by lemma 5. 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.

□