Information Flow in Secure Contexts*

Annalisa Bossi[1] Damiano Macedonio[1]
Carla Piazza[1,2] Sabina Rossi[1]

[1]Dipartimento di Informatica,
Università Ca' Foscari di Venezia

[2]Dipartimento di Matematica e Informatica,
Università degli Studi di Udine

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 stronger than necessary when some knowledge about the environment (context) in which the process is going to run is available. To relax this requirement 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 view of a process. As observation equivalence we consider the cases of weak 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.

*This work has been partially supported by the EU Contract IST-2001-32617 "Models and Types for Security in Mobile Distributed Systems" (MyThS) and the FIRB project RBAU018RCZ "Interpretazione astratta e model checking per la verifica di sistemi embedded".

To appear in Journal of Computer Security.

1 Introduction

The problem of protecting data in a multilevel system is one of the relevant issues in computer security. *Information flow security properties* have been proposed as a means to ensure confidentiality of classified information. These properties impose constraints on information flow among different groups of entities with different security levels. Often only two groups are considered and are labelled with the security levels *high* (H) and *low* (L). The condition is that no information should flow from H to L.

An early attempt to formalize the absence of information flow was the concept of *noninterference* proposed in the seminal paper by Goguen and Meseguer [11]. 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. Noninterference has been further developed in different settings such as programming languages [38, 36, 35, 3], trace models [20, 21], process calculi [30, 28, 33, 8, 6, 14], probabilistic models [2, 7], timed models [13], cryptographic protocols [1, 9, 4].

Noninterference aims at characterizing the complete absence of any information flow or, indeed stronger, the absence of any causal flow. As already noticed by many authors [29, 26, 27, 33, 16] this is too strong for practical applications. For instance, when two high level users communicate through an encrypted channel, a low level user may only know that a communication occurred. In this case there is a causal flow but not a (significant) information flow. More generally, there are situations referred to as *downgrading*, in which trusted entities are permitted to move information from high to low. Thus the policy requirements may admit restricted/controlled information flows. Sometimes it is more a question of functionality. Absolute noninterference can hardly ever be achieved in real systems. In realistic situations high level input interferes with low level output all the time [32]. Typically strict noninterference simply is not feasible due to clashes of resource which it demands. Consider a simple device that allows information flow from low to high but not from high to low. Such a device is feasible from a theoretical point of view only, in practice some causal flow from high to low is necessary to regulate the flow from low to high and avoid buffer overflow.

To deal with restricted/controlled information flows the notion of *intransitive noninterference* has been introduced (see [29, 26]). Flows from the high level to a trusted part and flows from the trusted part to the low level are admissible since the trusted part takes care of controlling them, while a direct flow from high to low is not allowed.

Total noninterference could be stronger than necessary also when some knowledge about the environment (context) in which the process is going to run is available. The following example illustrates one such situation. Consider a process representing a client of a bank using his card in an Automatic Teller Machine (ATM) to take money 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 money. All the actions involved concern the exchange of confidential (high level) information between the client and the bank. A *correct* ATM should read the codes, and if they are correct, it should give the money to the client. Since all the data are protected, no (high) information is revealed to an external observer; hence we can assume that the ATM context is secure for the 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 money and does not suspect the fraud. Clearly, this context is not secure for the 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.

In this paper we introduce the notion of *secure contexts for a class of processes* to generalize noninterference to manage the cases illustrated above. The notion of secure contexts for a class of processes is parametric with respect to both an observation equivalence relation and an operation used to characterize the low level view of a process. 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 [8]) 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, and we introduce contexts as particular SPA expressions. Secure contexts for a class of processes are introduced in Section 3. They are illustrated by means of examples. In Sections 4 and 5 we study two instances of our general definition through weak bisimulation and trace equivalence, respectively. In Section 6 we discuss some related works and show how downgrading can be modelled by means of secure contexts. Finally, in Section 7 we draw some conclusions.

2 Basic Notions

The *Security Process Algebra* (SPA) [8] is a variation of Milner's CCS [23], 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 $O = \{\overline{a}, \overline{b}, \ldots\}$ is a set of *output* actions; a special action τ which models internal computations, not visible outside the system; a complement function $\overline{\cdot}: \mathcal{L} \to \mathcal{L}$, such that $\overline{\overline{a}} = a$, for all $a \in \mathcal{L}$. $Act = \mathcal{L} \cup \{\tau\}$ is the set of all *actions*. Function $\overline{\cdot}$ is extended to Act by defining $\overline{\tau} = \tau$. 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 a renaming function such that $f(\overline{\alpha}) = \overline{f(\alpha)}$, $f(\tau) = \tau$, $f(H) \subseteq H \cup \{\tau\}$, 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 $(\mathcal{E}, Act, \rightarrow)$, whose states are processes, is defined by structural induction as the least relation generated by the axioms and inference rules reported in Table 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; $recZ.T[Z]$ is the recursive term which can perform all the actions of the term obtained by substituting $recZ.T[Z]$ to the place-holder Z in the context $T[Z]$.

To define security properties it is also useful to introduce the *hiding* operator, $/$, of CSP which can be defined as a relabelling as follows: for a given set $v \subseteq \mathcal{L}$, $E/v \equiv E[f_v]$ 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 $h.\mathbf{0}|(\ell.X + \tau.\mathbf{0})$ we can replace the variable X with the process $h.\mathbf{0}$ obtaining the process $h.\mathbf{0}|(\ell.h.\mathbf{0} + \tau.\mathbf{0})$; or we can replace X by the term $a.Y$ obtaining the term $h.\mathbf{0}|(\ell.a.Y + \tau.\mathbf{0})$. 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 $C[X] \equiv h.\mathbf{0}|(\ell.X + \tau.\mathbf{0})$ or $D[X] \equiv (\ell.X + \tau.\mathbf{0})|Y$ or $C'[X] \equiv Y|h.\mathbf{0}$. Hence, the notation $C[h.\mathbf{0}]$ stands for $h.\mathbf{0}|(\ell.h.\mathbf{0} + \tau.\mathbf{0})$, while $D[h.\mathbf{0}] \equiv (\ell.h.\mathbf{0} + \tau.\mathbf{0})|Y$ and $C'[h.\mathbf{0}] \equiv Y|h.\mathbf{0}$. 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 (e.g., if $C \equiv a.Z + b.W$ then $recZ.C[Z]$ and $recW.C[W]$ denote different terms).

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

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

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 $E \stackrel{t}{\Longrightarrow} E'$ 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, $E \stackrel{\hat{a}}{\Longrightarrow} E'$ stands for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\xrightarrow{\tau})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ means zero or more τ labelled transitions).

The *weak bisimulation* relation [23] 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 $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E',F') \in \mathcal{R}$;
- if $F \xrightarrow{a} F'$, 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 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 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 $Tr(E) = Tr(F)$.

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

Following [23] 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 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 weakly 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 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 extend the concept of noninterference by introducing a general notion of *secure contexts for a class of processes*. The idea is that a context represents the environment interacting with processes during their execution. For instance, in Figure 1 on the left we represent a database DB, containing both confidential and public information and running in a context which comprises a firewall, trojan horses and interfaces allowing the users to interact with the database. The security notion we intend to capture aims at ensuring that the interaction between the context and the database is transparent with respect to the high level information for low level users. This means that low level users cannot distinguish between the whole system (on the left of Figure 1) and the system where the database contains only low level information (on the right in Figure 1). As an immediate consequence we have that no confidential information of the database is revealed to the low level observers. Moreover, since the low level database cannot interact with high level users through the interfaces in the context, as a side effect we get that also the high level information contained in the context is not revealed.

The notion of *secure contexts for a class of processes* presented below 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

$$\text{for all } C[X] \in \mathcal{C} \text{ and for all } E \in \mathcal{P},\ C[E] \sim_l C[E_l].$$

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 $\mathcal{C} = \{\ell.X + \ell.Y + h.Y\}$, with $\ell \in L$ 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 $\ell.E + \ell.F + h.F \sim_l \ell.E_l + \ell.F + h.F$. 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 $\ell.F + \ell.E + h.E \sim_l \ell.F + \ell.E_l + h.E_l$. 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 $\ell.F + \ell.G + h.G \sim_l \ell.F + \ell.G + h.G$.

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 [11]. In the literature the low level view of a process is usually modelled using either *restriction* or *hiding* of high level actions. The first case corresponds to disallowing any external synchronization on high level actions; the second case simulates the situation in which all possible synchronizations are performed.

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. Then C_1 can be represented by a term of the form

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

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:

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

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* (see Section 6) 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 [25]. 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:

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

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

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

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

where the only low level action is the input CHECK. By observing that E_1 has checked a second time on the stock marked, a low level observer could be able to deduce that *Mr Earner*'s investments are bad. Hence, in this case the context representing *Mr Earner*'s machine is not secure with respect to E_1.

It is clear that in order to get a process such that *Mr Earner*'s machine is secure with respect to it, a CHECK action after the GOOD one should be added. However, with the process

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

the context is still not secure even if the only information which is revealed to the low level user is that a high level action has been performed but not which one. This is due to the fact that our security property is based on classical noninterference [11] and thus it disallows any direct or indirect flow of confidential information. To allow restrict information flow we need to opportunely redesign the processes. For instance, in this case it is sufficient to add the masking component $\text{CHECK}.\mathbf{0}$. The resulting program E_2

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

is now secure according to our definition. Its 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.

Another possibility to allow restricted flows is that of designing E_2 by using downgrading actions as described in Section 6.1.

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.

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.*

Proof. Let $\mathcal{C}_2$ be secure for $\mathcal{P}_2$ with respect to X. Since $\mathcal{P}_1 \subseteq \mathcal{P}_2$, then $\mathcal{C}_2$ is also secure for $\mathcal{P}_1$ with respect to X. Moreover, since $\mathcal{C}_1 \subseteq \mathcal{C}_2$, we get that $\mathcal{C}_1$ 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. In particular, we consider two observation equivalences, named weak bisimulation and trace equivalence. The choice of the observation equivalence clearly depends on the application of interest. In [9] the authors study security properties of cryptographic protocols based on noninterference and they discriminate between those properties for which trace equivalence is sufficient, e.g., authentication, secrecy, and integrity, and those properties for which deadlock-sensitive equivalences like bisimulation and testing equivalence are necessary, e.g., fairness and non-repudiation.

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

$$\text{for all } C[X] \in \mathcal{C} \text{ and for all } E \in \mathcal{P},\ C[E] \setminus H \approx_B C[E \setminus H] \setminus H.$$

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.

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

$C_1[E] \setminus H \equiv \tau.\overline{\text{PRICE\_LIST\_L}}.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$ 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 market 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 $C[E_2 \setminus H] \setminus H \equiv \text{CHECK.CHECK}.\mathbf{0}$, 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 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 [8]. 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 $E \in \mathcal{E}$. $E \in BNDC$ if and only if $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ for all contexts $C[X] \equiv X|\Pi$, with $\Pi \in \mathcal{E}_H$.*

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

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 $\mathcal{P}$

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

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.

- Each $F \in \mathcal{E}$ is secure for $\mathcal{P}$, since $F \setminus H \approx_B F \setminus H$.
- A variable is secure for $E \in \mathcal{P}$, since $E \setminus H \approx_B E \setminus H \setminus H$.
- 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 $C[X]$ is secure for $E \in \mathcal{P}$. If $C[E] \setminus H \xrightarrow{a} C'$, then there exists i such that $a = l_i \in L$, and $C' \equiv C_i[E] \setminus H$. So we have that $C[E \setminus H] \setminus H \xrightarrow{a} C_i[E \setminus H] \setminus H$, with $C_i[E] \setminus H \approx_B C_i[E \setminus H] \setminus H$, by hypothesis on $C_i[X]$. The case $C[E \setminus H] \setminus H \xrightarrow{a} C'$ is similar.
- Let $E \in \mathcal{P}$. If $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$, then $C[E] \setminus H \setminus v \approx_B C[E \setminus H] \setminus H \setminus v$, hence $C[E] \setminus v \setminus H \approx_B C[E \setminus H] \setminus v \setminus H$.
- Let $C[X]$ be secure for $E \in \mathcal{P}$ and consider $C[X][f]$, 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 $C' \equiv C''[f]$, there exists $b \in L \cup \{\tau\}$ such that $a = f(b)$ and $C[E] \setminus H \xrightarrow{b} C''$. Hence, $C[E \setminus H] \setminus H \stackrel{\hat{b}}{\Rightarrow} C''' \approx_B C''$, and $C[E \setminus H][f] \setminus H \stackrel{\hat{a}}{\Rightarrow} C'''[f] \approx_B C''[f]$. So $C[X][f]$ is secure for E.

□

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 $E \equiv h.\ell.\mathbf{0} + \overline{h}.\mathbf{0}$. 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 built 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 $a.Y_1 + b.Y_2 + c.Y_3$, 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. The only non trivial case is "Recursion". Let $C, D \in \mathcal{C}_s$ be weak bisimilar, we prove that $recY.C \approx_B recY.D$. Without loss of generality we assume $C[Y]$ and $D[Y]$ with at most the single free variable Y. The generalization 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 $\mathcal{S}$ on $\mathcal{C}_s$ as:

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

We prove $\mathcal{S}$ 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 $G[recY.D[Y]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q'$. The converse follows by the symmetry of $\mathcal{S}$.

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

Base. 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 $G' \in \mathcal{C}_s$. Also $G[recY.D[Y]] \equiv a.G'[recY.D[Y]] \xrightarrow{a} G'[recY.D[Y]]$ and $(G'[recY.C[Y]], G'[recY.D[Y]]) \in \mathcal{S}$.

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

- $G \in \mathcal{E}$. Trivially $G[recY.C[Y]] \equiv G[recY.D[Y]] \equiv G$.
- $G \equiv X$. Then $recY.C[Y] \xrightarrow{a} P$ by applying "Recursion" at last step. Therefore $C[recY.C[Y]] \xrightarrow{a} P$ with a shorter inference. By induction

  $$C[recY.D[Y]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q' \text{ with } (P, Q') \in \mathcal{S}.$$

  But $C[Y] \approx_B D[Y]$ implies $D[recY.D[Y]] \stackrel{\widehat{a}}{\Longrightarrow} Q'' \approx_B Q$. And we conclude

  $$recY.D[Y] \stackrel{\widehat{a}}{\Longrightarrow} Q''' \approx_B Q'' \approx_B Q \approx_B Q'$$

  since $D[recY.D[Y]] \approx_B recY.D[Y]$.
- $G \equiv \sum_i a_i.G_i$. Then $\sum_i a_i.G_i[recY.C[Y]] \xrightarrow{a} P$ by applying "Sum" at last step. So $a_i.G_i[recY.C[Y]] \xrightarrow{a} P$. Hence $P \equiv G_i[recY.C[Y]]$, with $G_i \in \mathcal{C}_s$. By "Sum", $G[recY.D[Y]] \xrightarrow{a} Q \equiv G_i[recY.D[Y]]$, and $(P, Q) \in \mathcal{S}$.

- $G \equiv G_1 \setminus v$. Then $G_1[recY.C[Y]] \setminus \xrightarrow{a} P$ by applying "Restriction" at 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

$$G_1[recY.D[Y]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q' \text{ with } (P', Q') \in \mathcal{S}.$$

  We conclude $G_1[recY.D[Y]] \setminus v \stackrel{\widehat{a}}{\Longrightarrow} Q \setminus v \approx_B Q' \setminus v$, with $(P, Q' \setminus v) \in \mathcal{S}$ 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 "Relabelling" at 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

$$G_1[recY.D[Y]] \stackrel{\widehat{a'}}{\Longrightarrow} Q \approx_B Q' \text{ with } (P', Q') \in \mathcal{S}.$$

  By construction of $\mathcal{S}$, we conclude

$$G_1[recY.D[Y]][f] \stackrel{\widehat{f(a')}}{\Longrightarrow} Q[f] \approx_B Q'[f] \text{ with } (P, Q'[f]) \in \mathcal{S}.$$

- $G \equiv recZ.G_1[X, Z]$. Then $recZ.G_1[recY.C[Y], Z] \xrightarrow{a} P$ by applying "Recursion" at last step. It derives by a shorter inference from

$$G_1[recY.C[Y], recZ.G_1[recY.C[Y], Z]] \xrightarrow{a} P.$$

  By induction we know that

$$G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \stackrel{\widehat{a}}{\Longrightarrow} Q \approx_B Q' \text{ with } (P, Q') \in \mathcal{S}.$$

  Since $G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \approx_B recZ.G_1[recY.D[Y], Z]$, we can finally conclude that

$$G_1[recY.D[Y], recZ.G_1[recY.D[Y], Z]] \stackrel{\widehat{a}}{\Longrightarrow} Q'' \approx_B Q \approx_B Q'.$$

□

Lemma 4.11. *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 assume C with at most the single free variable Y. The general case follows by Definition 2.5. Let $\mathcal{S}$ be defined as

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

If we prove $\mathcal{S}$ to be a strong bisimulation, then the Lemma 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.

In order to prove that $\mathcal{S}$ is a strong bisimulation, we are verifying that for any pair $(G[recY.(C \setminus H)] \setminus H, G[recY.C] \setminus H)$ in $\mathcal{S}$

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

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

We proceed by induction on the depth of the inference proof of $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ or $G[recY.C] \setminus H \xrightarrow{a} Q$.

Base. (1) If $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ with an inference of depth 1, then "Restriction" and "Prefix" have been applied. So, $G[X] \equiv a.G'[X]$ and $P \equiv G'[recY.(C \setminus H)] \setminus H$. By applying the same rules to $G[recY.C] \setminus H$ we obtain $G[recY.C] \setminus H \xrightarrow{a} Q \equiv G'[recY.C] \setminus H$ with $G' \in \mathcal{C}_S$. Hence $(P,Q) \in \mathcal{S}$. Case (2) is similar.

Induction step. We proceed by cases on the structure of $G[X]$. In (1) we consider $(recY.(C \setminus H)) \setminus H \xrightarrow{a} P$, and in (2) we consider $(recY.C) \setminus H \xrightarrow{a} Q$.

- $G[X] \in \mathcal{E}$. Trivial.
- $G[X] \equiv X$. (1)Then $P \equiv P' \setminus H$ and $C[recY.(C \setminus H)] \setminus H \xrightarrow{a} P'$ by a shorter inference. Hence P' is free from high level action, i.e. $P \equiv P' \setminus H \equiv P'$. By induction, $C[(recY.C)] \setminus H \xrightarrow{a} Q$, and so $(recY.C) \setminus H \xrightarrow{a} Q$, with $(P',Q) \in \mathcal{S}$. (2) Then $Q \equiv Q' \setminus H$ and $C[recY.C] \setminus H \xrightarrow{a} Q$ by a shorter inference. By induction $C[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$, and so $(recY.(C \setminus H)) \setminus H \xrightarrow{a} P$, with $(P,Q) \in \mathcal{S}$.
- $G[X] \equiv \sum_{i \in I} a_i.G_i[X]$. (1) Then there exists $i \in I$ such that $a \equiv a_i$ and $P \equiv G_i[recY.(C \setminus H)] \setminus H$. Hence, $G[recY.C] \setminus H \xrightarrow{a} Q$ with $Q \equiv G_i[recY.C] \setminus H$. From this we get $(P,Q) \in \mathcal{S}$. (2) Then $a \equiv a_i$ and $Q \equiv G_i[recY.C] \setminus H$. Therefore, $G[recY.(C \setminus H)] \setminus H \xrightarrow{a} P$ with $P \equiv G_i[recY.(C \setminus H)] \setminus H$, this means that $(P,Q) \in \mathcal{S}$.
- $G[X] \equiv G_1[X] \setminus v$. Trivial.
- $G[X] \equiv G_1[X][f]$. Trivial.
- $G[X] \equiv recZ.G_1[X,Z]$. (1) Then $recZ.G_1[recY.(C \setminus H),Z] \setminus H \xrightarrow{a} P$ and also $G_1[recY.(C \setminus H), recZ.G_1[recY.(C \setminus H),Z]] \setminus H \xrightarrow{a} P$ by a shorter inference. By induction $G_1[recY.C, recZ.G_1[recY.C,Z]] \setminus H \xrightarrow{a} Q$ and $(P,Q) \in \mathcal{S}$. Therefore, $recZ.G_1[recY.C,Z]] \setminus H \xrightarrow{a} Q$, i.e., $G[recY.C] \setminus H \xrightarrow{a} Q$. (2) Then it holds that $G_1[recY.C, recZ.G_1[recY.C,Z]] \setminus H \xrightarrow{a} Q$, by a shorter inference. By induction $G_1[recY.(C \setminus H), recZ.G_1[recY.(C \setminus H),Z]] \setminus H \xrightarrow{a} P$ with $(P,Q) \in \mathcal{S}$. Therefore, $recZ.G_1[recY.(C \setminus H),Z] \setminus H \xrightarrow{a} P$.

□

Lemma 4.12. *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 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 $\mathcal{C}_s$. By applying Lemma 4.11 to both members, we get $recY.(C[E]) \setminus H \approx_B recY.(C[E \setminus H]) \setminus H$, which is the thesis. □

Theorem 4.13. *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 4.12 we have the thesis.

□

EXAMPLE 4.14. 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

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

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 [8, 10, 5]). In particular, in [5] 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 [10]).

Definition 4.15 (P_BNDC). Let $E \in \mathcal{E}$. $E \in P\_BNDC$ if $E' \in BNDC$ for all E' reachable from E.

We will also use the following characterization of *P_BNDC* [5].

Theorem 4.16. *Let $E \in \mathcal{E}$ be a process. $E \in P\_BNDC$ iff for all E' reachable from E, if $E' \stackrel{h}{\rightarrow} E''$, then $E' \stackrel{\hat{\tau}}{\Longrightarrow} E'''$ and $E'' \setminus H \approx_B E''' \setminus H$.*

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 4.17. *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. Consider the following 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 on high actions. Assume that $(E|G) \setminus H \stackrel{\tau}{\rightarrow} (E'|G') \setminus H$ with $E \stackrel{h}{\rightarrow} E'$ and $G \stackrel{\bar{h}}{\rightarrow} G'$. Since $E,G \in P\_BNDC$, by Theorem 4.16 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 $G' \setminus H \approx_B G'' \setminus H$. So, $E \setminus H \stackrel{\hat{\tau}}{\Rightarrow} E'' \setminus H$ 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. $((E'|G') \setminus H, (F'|K') \setminus H) \in \mathcal{S}$. □

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.

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

Definition 4.18 ($\mathcal{P}$-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 4.19 ($\mathcal{P}$-secure contexts). 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 4.20. *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* (see [10]).

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*. We get the thesis, by applying Lemma 4.17 to the four processes. □

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 4.20 are secure also for the subclasses of *P_BNDC* named *SBNDC* (see [8]), *PP_BNDC*, and *CP_BNDC* (see [5]), respectively.

EXAMPLE 4.21. Consider the programs E_2 and E_3 of Example 3.4. They are *P_BNDC*, hence by applying Theorem 4.20 we immediately get that the two contexts of Example 3.4 are secure for these processes.

EXAMPLE 4.22. Let $\text{END} \in \mathcal{L}$ be an action and E be a *P_BNDC* process in which neither END nor $\overline{\text{END}}$ occur. Let $\mathcal{P}_{\text{END}}$ be a class of *P_BNDC* processes whose termination is announced by the execution of an END action. Consider the context C defined as

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

When in C we replace the variable X with a process F taken from $\mathcal{P}_{\text{END}}$ 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 4.20 and Proposition 3.5, we have that $X|\overline{\text{END}}.E$ is secure for $\mathcal{P}_{\text{END}}$. Hence, from Theorem 4.6, we obtain that C is secure for $\mathcal{P}_{\text{END}}$.

Theorem 4.20 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 4.23 (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 4.24. The contexts X and $W \setminus H$ belong to $\mathcal{C}_p$. Hence, by using the constructor $\ell.Z_1 + h.Y_1 + \tau.Y_1$, the context $\ell.(W \setminus H) + h.X + \tau.X$ belongs to $\mathcal{C}_p$.

Theorem 4.25. *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 [5], 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 [10]).

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 4.17. □

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

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

If we consider the context C_1, that is $X|\overline{\text{PWD\_SHOPKEEPER}}.\mathbf{0}$, we have $C_1[E] \setminus H \equiv \tau.\mathbf{0} + \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$ is not weakly bisimilar to $C_1[E \setminus H] \setminus H \equiv \overline{\text{PROD\_LIST\_L}}.\mathbf{0}$. 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. This example recalls the work presented in [37] where the authors claim the need to define properties in terms of sequences of interactions (traces) between the system and the users.

In this section we consider the following instance of our security definition: $\sim$ is $\approx_T$ (trace equivalence) and $\cdot_I$ 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

$$\text{for all } C[X] \in \mathcal{C} \text{ and for all } E \in \mathcal{P}, C[E] \setminus H \approx_T C[E \setminus H] \setminus H.$$

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 [8]) 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 [8].

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. ($\Rightarrow$) 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 $E \setminus H \setminus H \approx_T E \setminus H$, and then $(E \setminus H|\Pi) \approx_T E \setminus H$. Hence $(E|Pi) \setminus H \approx_T (E \setminus H|\Pi) \setminus H$, by transitivity of $\approx_T$.

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

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 $C[E] \setminus H \approx_B C[E \setminus H] \setminus H$ 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. The 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. 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 $C_i[E] \setminus H \approx_T C_i[E \setminus H] \setminus H$. We proceed by exploiting the fact that $\approx_T$ is a congruence with respect to the non deterministic choice operator, and the restriction operator commutes with the non deterministic choice. Hence we obtain

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

and so $(\sum_{i \in I} C_i[E]) \setminus H \approx_T (\sum_{i \in I} C_i[E \setminus H]) \setminus H$.

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 $(\sum_{i \in I} C_i[E] + \sum_{h_j \in H} h_j.D_j[E]) \setminus H \approx_T (\sum_{i \in I} C_i[E \setminus H] + \sum_{h_j \in H} h_j.D_j[E \setminus H]) \setminus H$, i.e. our thesis. □

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 analogues of the results proved in Subsection 4.2 for *P_BNDC* processes, in the case of *NDC* processes. In particular, the following lemma corresponds to Lemma 4.17.

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. The following points are proved by Focardi and Gorrieri:

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

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

(3) 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 Lemma 5.4
$(E\|G)/H$	$\approx_T$	by (2)
$(E/H\|G/H)$	$\approx_T$	by Lemma 5.4 and (3)
$(F/H\|K/H)$	$\approx_T$	by (2)
$(F\|K)/H$	$\approx_T$	by (1) and Lemma 5.4
$(F\|K) \setminus H$.		

□

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. We recall that, by Definition 4.19, 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 5.9. *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*. We get the thesis by applying Lemma 5.8 to these four processes. □

Theorem 5.9 does not provide a decidability result. In the following definition we characterize a decidable class of *NDC*-contexts, which is the analogue of the class $\mathcal{C}_p$ of Definition 4.23.

Definition 5.10 (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: $\ell.Y$ with $\ell \in L$, $Y|Z$, $Y \setminus v$, $Y[f]$, $Y + Z$, $h.Y + \tau.Y$ with $h \in H$.

Theorem 5.11. *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 $E|F, E \setminus H \in NDC$.

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 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 for all $E \in NDC$:

$$\begin{aligned} Tr((C[E] + D[E]) \setminus H) &= Tr((C[E] \setminus H) + (D[E] \setminus H)) \\ &= Tr(C[E] \setminus H) \cup Tr(D[E] \setminus H) \\ &= Tr(C[E \setminus H] \setminus H) \cup Tr(D[E \setminus H] \setminus H) \\ &= Tr(C[E \setminus H] + D[E \setminus H]) \end{aligned}$$

so we conclude that $C[X] + D[X]$ is secure for *NDC*. □

6 Related Works

Since the seminal work by Goguen and Meseguer [11], noninterference has played 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 indeed no real policy ever calls for total absence of information flow over any channel. In many practical applications confidential data can flow from high to low provided that the flow is not direct and it is controlled by the system, i.e., a trusted part of the system

can control the downgrading of high level information. Consider for instance the case in which the high level user edits a file and sends it through a private channel to an encrypting protocol, the encrypting protocol encrypts the file and sends it using a public channel. Even if the high level data are sent using a public channel the fact that the file is encrypted ensures that the low level users cannot read the data. In fact, the low level users can only observe that an encrypted file is passing on the public channel. In this case the encrypting protocol represents the trusted part of the system which controls the flow from high to low.

The problem of detecting only uncontrolled information flows has first been considered by Goguen and Meseguer in [12]. They introduce the notion of *conditional noninterference* which admits flow from high to low level through a controlled channel. Rushby in [29] develops a theory of downgrading in the deterministic case based on the notion of *intransitive noninterference*. Pinsky in [26] unifies the concepts of standard and intransitive noninterference and describes a decision procedure for noninterference. In [27] a formalization of intransitive noninterference in the context of deterministic CSP is presented. In [33] the relationships between various definitions of noninterference and notions of process equivalence are analyzed and some generalizations to handle *partial* and conditional information flows are outlined. The authors provide a general definition of noninterference and 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 [31, 33] and generalizes the formalization presented in those papers by allowing the use of more structured contexts and not considering only trace-based equivalences.

Another approach to the problem of achieving noninterference in real systems is presented in [7] where a probabilistic framework is used to give a quantitative estimate of the information flowing through the systems. The authors use a parameterized behavioral equivalence to consider as effectively noninterfering two distinguishable behaviors provided that their difference is below a threshold ε. We can handle this idea just instantiating our notion of secure context with their parameterized behavioral equivalence. The presence of contexts in our definition allows the treatment of cases in which the similarity between two processes strongly depends on the environment in which they evolve.

In [17, 18], 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 by Martinelli consists of checking that, for any instance of the unknown component, the resulting system satisfies a property expressed as a formula of a suitable temporal logic. In order to make decidable the verification problem, he does not consider constructs for modelling recursion. He also studies a method for finding, if it exists, a suitable system to be inserted into an unspecified component so that the whole system respects a given specification. In [17], it is also proposed a generalization of Focardi and Gorrieri's *Non Deducibility on Composition* (*NDC* and *BNDC*) by parameterizing the equivalence relation over processes. In our work we endorse this idea of generalizing *NDC* and we extend it by parameterizing also the power of an external observer (by introducing the

concept of low level view) and the power of a generic attacker (by introducing the context).

Secure contexts are also studied by Sabelfeld and Mantel in [34] 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 [19] 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 [22, 39, 36]). As we point out in the previous sections there is 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 4.20).

In [24] *admissible interference* (*AI*) is introduced as a trace based generalization of *SNNI* [8] to deal with downgrading. In [15] a bisimulation based version of *AI*, named *BNAI*, is presented and applied to the analysis of cryptographic protocols. Like in our approach, their basic model is a variant of CCS. This facilitates the comparison with our work as shown below.

6.1 Persistent Secure Contexts and Downgrading

In order to model the notion of downgrading in our language we need to introduce the set of actions performed by the trusted downgrader, i.e., we assume that $\mathcal{L}$ is partitioned into the sets D (downgrading actions), H, and L. In the following we denote by H^+ the set $H \cup D$. It is reasonable to assume that an attacker cannot simulate the trusted part of the system, i.e., it cannot perform the actions in D. For instance, in the case of protocol analysis the attacker cannot distribute the encryption keys. Moreover, we can assume that the low level users cannot observe the actions performed by the trusted part. These considerations can be translated in our framework as follows:

- the class $\mathcal{C}$ of contexts in which we are interested has to be a subset of the set $\mathcal{C}_H$ of all contexts built using only actions in H;
- the operation $\cdot_l$ has to remove all the behaviors relative to actions in H^+.

In particular, if we consider our first instance, i.e., using weak bisimulation and restriction, and we focus on the class of contexts $\mathcal{C}_{BNDC} = \{X|\Pi \,|\, \Pi \in \mathcal{E}_H\}$ (i.e, the contexts used to define *BNDC*) we get that a process E has to satisfy

$$(E|\Pi) \setminus H^+ \approx_B (E \setminus H^+ \,|\Pi) \setminus H^+$$

for all $\Pi \in \mathcal{E}_H$.

EXAMPLE 6.1. Let us consider the case in which an encrypting protocol receives a confidential file on a private channel, encrypts it and sends the resulting file on a public channel. Let $file_h$ be the high level input representing the reception of the file on the private channel, enc_d be the downgrading action representing the encryption phase, $\overline{ok_h}$

be a confidential acknowledge to the high level user, and $\overline{file_l}$ be the low level output of the encrypted data. The encrypting protocol can be formalized as

$$Enc \equiv file_h.enc_d.\overline{ok_h}.\overline{file_l}.\mathbf{0}$$

Since it is reasonable to assume that an attacker cannot simulate the trusted part of the system, i.e., it cannot perform the actions in D, if we consider any possible attacker $\Pi \in \mathcal{E}_H$ we get that

$$(Enc|\Pi) \setminus H^+ \approx_B \mathbf{0} \approx_B (E \setminus H^+|\Pi) \setminus H^+$$

which means that Enc is secure.

Unfortunately imposing that E satisfies

$$(E|\Pi) \setminus H^+ \approx_B (E \setminus H^+|\Pi) \setminus H^+$$

is not enough to guarantee no information flow. In fact, all the (uncontrolled) flows which occur after the first downgrading are not revealed. This problem was observed also in [24]. As done in [24, 15] we can check the flows occurring after the first downgrading by imposing *persistency*.

Definition 6.2 (Persistent-secure Contexts for a Process). A class of contexts $\mathcal{C}$ is *persistent-secure* for a process E iff for all E' reachable from E, $\mathcal{C}$ is secure for E'.

Applying this definition to weak bisimulation and restriction with respect to H^+, and considering the class of contexts $\mathcal{C}_{BNDC}$ we get that a process E has to be such that for all E' reachable from E it holds

$$(E'|\Pi) \setminus H^+ \approx_B (E' \setminus H^+|\Pi) \setminus H^+$$

EXAMPLE 6.3. Let us consider again the encrypting protocol Enc above, it reaches the process $E' \equiv \overline{ok_h}.\overline{file_l}.\mathbf{0}$ which does not satisfy

$$(E'|\Pi) \setminus H^+ \approx_B (E' \setminus H^+|\Pi) \setminus H^+$$

In fact if $\Pi \equiv ok_h.\mathbf{0}$ then $(E'|\Pi) \setminus H^+ \approx_B \overline{file_l}.\mathbf{0}$, while $(E' \setminus H^+|\Pi) \setminus H^+ \approx_B \mathbf{0}$. This means that the low level user which observes the encrypted file passing on the public channel can infer that the high level user has received the acknowledge. We can avoid this kind of flow by adding a timeout to the protocol

$$Enc \equiv file_h.enc_d.(\overline{ok_h}.\overline{file_l}.\mathbf{0} + \tau.\overline{file_l}.\mathbf{0}).$$

Now the process is secure.

The *BNAI* property introduced in [15] corresponds to consider $\cdot_l$ equal to $\setminus H^+$, $\sim$ equal to $\approx_B$, and the class of contexts $\mathcal{C}_{BNAI}$ of the form $(X \setminus D)/H$. A process E is *BNAI* if and only if $\mathcal{C}_{BNAI}$ is persistent-secure for E. We can prove that this is equivalent to consider the class of contexts $\mathcal{C}_{BNDC}$.

7 Conclusions

We presented a generalization of the notions of noninterference which is more flexible than the ones introduced by Focardi and Gorrieri [8]. 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 occurs in many practical applications. On the other hand our notion allows us to enlarge the set of possible attackers, since contexts can also perform low level actions and SPA operators can be freely combined in the context construction.

As noted by other authors (see, e.g., [8, 33, 17]) the notion of noninterference strongly depends on the notion of process equivalence. But the problem of characterizing the behavioral equality between two processes is not trivial in a non-deterministic system. In fact, there is no notion of system equivalence which everybody agrees upon, the choice of the appropriate notion of equivalence depends on the environment and application which are considered. The equivalence can be chosen among, for example, trace or failure equivalence, various forms of bisimulation and testing equivalence. Our notion is parametric with respect to the relational equivalence among processes, hence it can be specified in order to fit the right idea of process equality in various contexts of study.

In modelling real systems we cannot ignore the abilities of the low level observer. This is captured in our approach by parameterizing also the low level view in order to fit the situation in the real systems.

References

[1] Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, 1999.

[2] A. Aldini, M. Bravetti, and R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Non-interference. *Journal of Computer Security*, 2004. To appear.

[3] V. Benzaken, M. Burelle, and G. Castagna. Information flow security for xml transformations. In *Proc. of Asian Computing Science Conference (ASIAN'03)*, LNCS. Springer-Verlag, 2003. To appear.

[4] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for secrecy and non-interference in networks of processes. In Victor E. Malyshkin, editor, *Proc. of International Conference on Parallel Computing Technologies*, volume 2127 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, 2001.

[5] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Verifying Persistent Security Properties. *Computer Languages, Systems and Structures*, 2004. To appear. Available at `http://www.dsi.unive.it/~srossi/cl04.ps`.

[6] 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.

[7] A. Di Pierro, C. Hankin, and H.Wiklicky. Approximate Non-Interference. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 3–17. IEEE Computer Society Press, 2002.

[8] R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Proc. of Foundations of Security Analysis and Design (FOSAD'01)*, volume 2171 of *LNCS*, pages 331–396. Springer-Verlag, 2001.

[9] 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.

[10] 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.

[11] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'82)*, pages 11–20. IEEE Computer Society Press, 1982.

[12] J. A. Goguen and J. Meseguer. Unwinding and Inference Control. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'84)*, pages 75–86. IEEE Computer Society Press, 1984.

[13] R. Gorrieri, E. Locatelli, and F. Martinelli. A simple language for real-time cryptographic protocol analysis. In P. Degano, editor, *Proc. of European Symposium on Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 114–128. Springer-Verlag, 2003.

[14] 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.

[15] S. Lafrance and J. Mullins. Bisimulation-based Non-deterministic Admissible Interference and its Application to the Analysis of Cryptographic Protocols. *Electronic Notes in Theoretical Computer Science*, 61:1–24, 2002.

[16] G. Lowe. Quantifying Information Flow. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 18–31. IEEE Computer Society Press, 2002.

[17] F. Martinelli. *Formal Methods for the Analysis of Open Systems with Applications to Security Properties*. Phd thesis, University of Siena, December 1999.

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

[19] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'87)*, pages 161–166. IEEE Computer Society Press, 1987.

[20] J. McLean. Security Models and Information Flow. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'90)*, pages 180–187. IEEE Computer Society Press, 1990.

[21] J. McLean. Security Models. *Encyclopedia of Software Engineering*, 1994.

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

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

[24] J. Mullins. Nondeterministic Admissible Interference. *Journal of Universal Computer Science*, 11:1054–1070, 2000.

[25] C. O'Halloran. A Calculus of Information Flow. In *Proc. of the European Symposium on Research in Security and Privacy (ESoRiCS'90)*, pages 180–187. AFCET, 1990.

[26] S. Pinsky. Absorbing Covers and Intransitive Noninterference. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'95)*, pages 102–113. IEEE Computer Society Press, 1995.

[27] A. W. Roscoe and M. H. Goldsmith. What is intransitive noninterference? In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 228–238. IEEE Computer Society Press, 1999.

[28] A. W. Roscoe, J. C. P. Woodcock, and L. Wulf. Non-Interference through Determinism. *Journal of Computer Security*, 4(1), 1996.

[29] J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL-92-02, SRI International, December 1992.

[30] P. Y. A. Ryan. A CSP Formulation of Non-Interference and Unwinding. *Cipher*, pages 19–27, 1991.

[31] P.Y.A Ryan. Mathematical Models of Computer Security. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*, pages 1–62. Springer-Verlag, 2001.

[32] P.Y.A. Ryan, J. McLean, J. Millen, and V. Gilgor. Non-interference, Who Needs It? In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 237–238. IEEE Computer Society Press, 2001.

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

[34] A. Sabelfeld and H. Mantel. Static Confidentiality Enforcement for Distributed Programs. In M. V. Hermenegildo and G. Puebla, editors, *Proc. of Int. Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 376–394. Springer-Verlag, 2002.

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

[36] A. Sabelfeld and D. Sands. Probabilistic Noninterference for Multi-threaded Programs. In *Proc. of the IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 200–215. IEEE Computer Society Press, 2000.

[37] S. Schneider and A. Sidiropoulos. CSP and Anonymity. In *European Symposium on Research in Computer Security (ESORICS'96)*, volume 1146 of *LNCS*, pages 198–218. Springer-Verlag, 1996.

[38] 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.

[39] A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *Proc. of the IEEE Symposium on Security and Privacy (SSP'97)*, pages 74–102. IEEE Computer Society Press, 1997.

Prefix	$\dfrac{-}{a.E \xrightarrow{a} E}$
Sum	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 + E_2 \xrightarrow{a} E_1'} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 + E_2 \xrightarrow{a} E_2'}$
Parallel	$\dfrac{E_1 \xrightarrow{a} E_1'}{E_1 \mid E_2 \xrightarrow{a} E_1' \mid E_2} \quad \dfrac{E_2 \xrightarrow{a} E_2'}{E_1 \mid E_2 \xrightarrow{a} E_1 \mid E_2'} \quad \dfrac{E_1 \xrightarrow{\ell} E_1' \;\; E_2 \xrightarrow{\bar{\ell}} E_2'}{E_1 \mid E_2 \xrightarrow{\tau} E_1' \mid E_2'}$
Restriction	$\dfrac{E \xrightarrow{a} E'}{E \setminus v \xrightarrow{a} E' \setminus v}$ if $a \notin v$
Relabelling	$\dfrac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$
Recursion	$\dfrac{T[recZ.T[Z]] \xrightarrow{a} E'}{recZ.T[Z] \xrightarrow{a} E'}$

Table 1: The operational rules for SPA

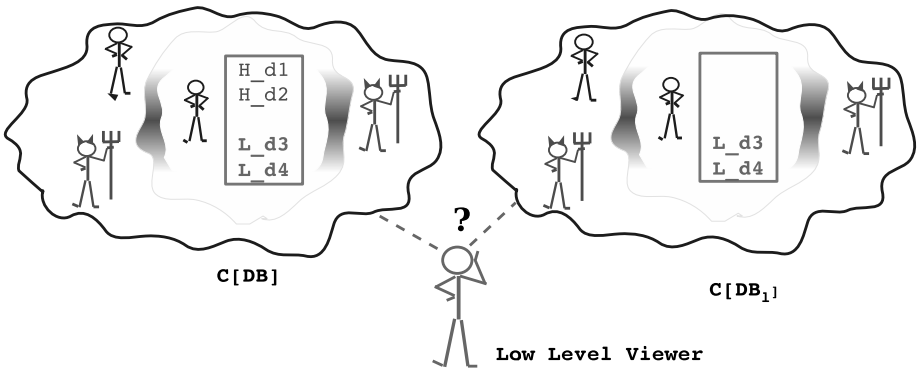

Figure 1: The Database example.