A Uniform Approach for the Definition of Security Properties*

Riccardo Focardi[1] and Fabio Martinelli[2]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy.
focardi@dsi.unive.it

[2] Dipartimento di Informatica, Università di Pisa, Italy.
martinel@di.unipi.it

Abstract. We present a uniform approach for the definition and the analysis of various security properties. It is based on the general idea that a security property should be satisfied even in the presence of an hostile environment. This principle determines a family of strong properties which are resistant to every external attack, but are quite impractical to check. For this reason, we find some general conditions that permit to check a property only against a "most powerful" intruder. We show that the results of our theory can be easily applied to a number of existing security properties that can be rephrased in our setting. This shows the generality of the approach and permits to find some interesting relations among properties which have been proposed for different security issues.

1 Introduction

In the last years, there has been an increasing diffusion of distributed systems where resources and data are shared among users located everywhere in the world. It is clear that the use of wide area networks highly increases the possibility of intrusions in a system. Moreover, it is likely that a user downloads some malicious programs from an untrusted source on the net and executes them inside its own system with unpredictable results. Moreover, it is possible that a system is completely secure inside but adopts weak mechanisms for remote connections. The situation above becomes crucial if, for example, we want to use the network for some critical activity such as electronic commerce or home banking.

All of these arguments and many others have recently focused the attention of many researches on the study of security issues. One of the most interesting challenges is to find a way of guaranteeing that a certain security policy, protocol or mechanism reaches the aim for which it was designed. For this reason, the interest on formal methods for the specification and analysis of security properties has recently enormously increased (see, e.g., [1, 2, 4, 8, 9, 10, 11, 13, 15, 17, 18, 21, 22, 23, 26]). Typically, we would like to be guaranteed that a certain

* Work partially supported by CNR, Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria" and by MURST Progetto "Tecniche Formali per la Specifica, l'Analisi, la Verifica e la Trasformazione dei Sistemi Software".

security property holds in every possible hostile environment. As an example, if we want to check that a certain protocol guarantees "mutual authentication", we have to check that *every possible intruder* is not able to impersonate one of the two parties in the communication.

In this work, we propose a general schema for the definition of security properties which is actually based on this idea of checking a system against all the possible hostile processes. Such an idea was already studied in [8, 9, 18] for a particular property called *Non Deducibility on Compositions* (NDC, for short). Our general schema is a generalization of NDC which has the following form:

$$E \text{ satisfies } P_{\lhd}^{\alpha} \qquad \textit{iff} \qquad \forall X \in \mathit{Env} : E \,\|\, X \lhd \alpha(E) \tag{1}$$

Basically, the general property $P_{\lhd}^{\alpha}$ requires that the system E satisfies a specification $\alpha(E)$ when composed (in parallel) with any (possibly hostile) environment X. The property is parametric with respect to $\alpha(E)$ and $\lhd$ that can be instantiated in order to obtain different security properties. In particular $\alpha(E)$ is a function between processes that, given E, specifies which should be the "correct" (intended) behaviour of E; $\lhd$ is a relation between processes that represents our notion of "observation". Thus, with $E \,\|\, X \lhd \alpha(E)$ we actually check if process E shows a correct behaviour even in the presence of X.

This universal quantification over all the possible intruders could be problematic when trying to check a property, since we have to verify it over infinitely many processes (one for each intruder). Usually, this problem is overcome by analyzing the case where only the "most powerful" intruder is considered (e.g., see [6, 14, 24]). As a matter of fact, if the property holds in the presence of the most powerful intruder then it will certainly hold even if we consider less powerful ones. However, it could be not always the case that such most general process exists, and if so, it could be non trivial to define it.

We formally study this, by giving some conditions on $\lhd$ that permit to statically characterize a property $P_{\lhd}^{\alpha}$, i.e., that permit to "reduce" it to the case where only the most general intruder is considered. For some interesting properties (e.g., BNDC [9]) it seems very difficult to find such a reduction. On the other hand, when such a reduction can be applied we obtain a formal proof that, for these properties, the "most powerful" intruder approach is indeed correct.

We show the generality of the approach by rephrasing a number of existing properties as particular instances of (1). For example, we can define *NDC*, *BNDC* [9, 10], *Agreement* [15], *authentication* as proposed in [26] and *non-repudiation* [25] by choosing particular instances of $\lhd$ and $\alpha(E)$. For many of them, the $\lhd$ relation corresponds to *trace preorder* which can be shown to satisfy the conditions for the static characterization, thus permitting to "safely" apply the most general intruder approach for the analysis of such properties.

Our general schema permits also to study relationships among different security properties. Indeed, if we rephrase these properties in our model, their comparison can be carried out by simply studying the relations among the relative α's and $\lhd$'s. Some of the properties we consider have been proposed for completely different aims. For example, NDC has been introduced for studying

Non-Interference [11] in non-deterministic systems while *Agreement* has been proposed for the analysis of entity authentication in protocols. This provides some interesting results, e.g., we show how Non-Interference properties can be used to guarantee authentication, and suggests the possibility of finding a general taxonomy of security properties.

The paper is organized as follows. In Section 2 we define a variant of value-passing CCS [20] which has been provided with auxiliary operators for the data handling and the modelling of cryptography. Then, in Section 3, we define our general schema and we give some general results on it. Section 4 is about the rephrasing of some existing definitions in our model that also permits to give a formal comparison of them. Finally, in Section 5 we give some concluding remarks and discuss future work.

2 The Model

In this section we present the language we will use for the specification of security properties and protocols. It is basically a variant of value-passing CCS [20], where we explicitly give to processes the possibility of manipulating messages. In particular, processes may perform message encryption and decryption, moreover they can also construct "complex" messages by composing together simple ones. Since we are generalizing the idea of NDC [9], we have actually decided to use (an extension of) the language adopted for the definition of such property. However, the ideas we propose in the following could be naturally defined also in other languages, e.g., CSP [12, 23] and spi-calculus [1, 2].

2.1 The Language Syntax

In this section we give the syntax of the language. As mentioned above, it is an extension of CCS and in particular of the *Security Process Algebra* proposed in [10] for the description and the analysis of multi-level systems. This extension borrows some concepts from the language proposed in [17] for the analysis of authentication protocols. We call the calculus *Cryptographic Security Process Algebra* (CryptoSPA for short). Its syntax is based on the following elements:

- A set $I = \{a, b, \ldots\}$ of *input* channels, a set $O = \{\bar{a}, \bar{b}, \ldots\}$ of *output* ones;
- A set M of basic messages and a set K of encryption keys with a function $\cdot^{-1} : K \to K$ such that $(k^{-1})^{-1} = k$. The set $\mathcal{M}$ of all messages is defined as the least set such that $M \cup K \in \mathcal{M}$ and $\forall m \in \mathcal{M}$, $\forall k \in K$ we have that (m, m') and $\{m\}_k$ also belong to $\mathcal{M}$. Basically, $\mathcal{M}$ is obtained by applying inductively the pair message constructor (m, m') and the encryption one $\{m\}_k$, starting from the set $M \cup K$ of basic messages and keys;
- A family $\mathcal{U}$ of sets of messages and a function $Msg(c) : I \cup O \longrightarrow \mathcal{U}$ which maps every channel c into the set of possible messages that can be sent and received on such channel. Msg is such that $Msg(c) = Msg(\bar{c})$.
- A set C of *public* channels; these channels represent the insecure network where the enemy can intercept and fake messages;

- A set $Act = \{c(m) \mid m \in Msg(c)\} \cup \{\overline{c}\,m \mid m \in Msg(c)\} \cup \{\tau\}$ of actions (τ is the internal, invisible action), ranged over by a; we also have a function $chan(a)$ which returns c if a is either $c(m)$ or $\overline{c}\,m$, and the special channel $void$ when $a = \tau$; we assume that $void$ is never used within a restriction operator (see below).
- A set $Const$ of constants, ranged over by A.

The syntax of CryptoSPA agents is defined as follows:

$$E ::= \underline{0} \;\Big|\; c(x).E \;\Big|\; \overline{c}\,e.E \;\Big|\; \tau.E \;\Big|\; E+E \;\Big|\; E \,\|\, E \;\Big|\; E \setminus L \;\Big|\; E[f] \;\Big|$$
$$\Big|\; A(m_1, \ldots, m_n) \;\Big|\; [e = e']E; E \;\Big|\; [\langle e_1 \ldots e_r \rangle \vdash_{rule} x]E; E$$

where x is a variable, $m_1, \ldots, m_n$ are messages and $e, e_1, \ldots, e_r$ are variables or messages and L is a set of input channels. Both the operators $c(x).E$ and $[\langle e_1 \ldots e_r \rangle \vdash_{rule} x]E; E'$ bind variable x in E. It is also necessary to define constants as follows: $A(x_1, \ldots, x_n) \stackrel{\mathrm{def}}{=} E$ where E is a CryptoSPA agent which may contain no free variables except $x_1, \ldots, x_n$, which must be distinct.

Intuitively, $\underline{0}$ is the empty process that can do nothing; $c(x).E$ represents the process that can get an input m on channel c behaving like E where all the occurrences of x are substituted with m (written $E[m/x]$); $\overline{c}\,m.E$ is the process that can send m on channel c then behaving like E does; process $\tau.E$ can execute the internal (invisible) action τ and, after that, it behaves like E; the $E_1 + E_2$ process represent the non-deterministic choice which can behave either like E_1 or E_2; [1] $E_1 \,\|\, E_2$ is the parallel composition of E_1 and E_2, where the executions of E_1 and E_2 are interleaved and the two processes can communicate by synchronizing on complementary input/output actions, producing a τ action; as an example of communication, consider system $c(x).E_1 \,\|\, \overline{c}\,m.E_2$ which can execute an internal τ action moving to $E_1[m/x] \,\|\, E_2$; process $E \setminus L$ cannot send and receive messages on channels in L, for all the other channels it behaves exactly like E; in $E[f]$ every channel c of E is relabelled into $f(c)$; $A(m_1, \ldots, m_n)$ behaves like the respective definition where all the variables $x_1, \ldots, x_n$ are substituted with messages $m_1, \ldots, m_n$; the $[m = m']E_1; E_2$ process behaves as E_1 if $m = m'$ and as E_2 otherwise. [2]

The operators described so far are the standard value-passing CCS ones. We have an additional operator that has been introduced in order to model message handling and cryptography. Informally, the $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2$ process tries to deduce an information z from the tuple of messages $\langle m_1 \ldots m_r \rangle$ through one application of rule $\vdash_{rule}$; if it succeeds then it behaves like $E_1[z/x]$, otherwise it behaves like E_2; for example, given a rule $\vdash_{dec}$ for decryption, process $[\langle \{m\}_k, k^{-1} \rangle \vdash_{dec} x]E_1; E_2$ decrypts message $\{m\}_k$ through key k^{-1} and behaves like $E_1[m/x]$ while $[\langle \{m\}_k, k' \rangle \vdash_{dec} x]E_1; E_2$ (with $k' \neq k^{-1}$) tries to decrypt the

[1] We sometimes write $\sum$ (indexed on a set) to represent a n-ary/infinitary sum.

[2] Here we consider syntactic equality among messages, however other definitions may be given depending on the algebraic features of the cryptographic system which is assumed (e.g., see [24]).

$$\frac{m \quad m'}{(m,m')}\ (\vdash_{pair}) \qquad \frac{(m,m')}{m}\ (\vdash_{fst}) \qquad \frac{(m,m')}{m'}\ (\vdash_{snd})$$

$$\frac{m \quad k}{\{m\}_k}\ (\vdash_{enc}) \qquad \frac{\{m\}_k \quad k^{-1}}{m}\ (\vdash_{dec})$$

Fig. 1. Inference System for message manipulation, where $m, m' \in \mathcal{M}$ and $k, k^{-1} \in K$.

same message with the wrong inverse key k' and (since it is not permitted by $\vdash_{dec}$) it behaves like E_2.

We call $\mathcal{E}$ the set of all the CryptoSPA terms, and we define $sort(E)$ to be the set of all the channels syntactically occurring in the term E. In the next section, we give the formal semantics of CryptoSPA together with the deduction rules $\vdash_{rule}$ for message manipulation.

2.2 The Operational Semantics of CryptoSPA

The semantics of CryptoSPA terms is given through labelled transition systems. A labelled transition system is essentially an automaton although it may have infinitely many states. Labelled transition systems constitute the preferred semantic domain for the operational description of many concurrent languages.

Definition 1. *A* labelled transition system (LTS) *is a triple* $(S, T, \rightarrow)$ *such that: S is a set of states, T is a set of labels (actions) and $\rightarrow \subseteq S \times T \times S$ is a set of labelled transitions.* ■

$(S_1, \alpha, S_2) \in \rightarrow$ (or equivalently $S_1 \xrightarrow{\alpha} S_2$) means that the system can move from the state S_1 to the state S_2 through the action α.

In order to model message handling and cryptography, we define an inference system which formalizes the way messages may be manipulated by processes. We say that m is *deducible* from a set of messages ϕ (and write $\phi \vdash m$) if there exists a proof of m whose leaves have premises contained in ϕ. Each inference system induces a *deduction function* $\mathcal{D}(\phi) = \{m \mid \phi \vdash m\}$. In Figure 1, we present a formalization (as inference system) of a simple deduction system which is indeed quite similar to those used by many authors (see, e.g., [14, 16]). It encodes all the operations a process can do over messages besides communicating them. In particular it can combine two messages obtaining a pair (rule $\vdash_{pair}$); it can extract one message from a pair (rules $\vdash_{fst}$ and $\vdash_{snd}$); it can encrypt a message m with a key k obtaining $\{m\}_k$ and finally decrypt a message of the form $\{m\}_k$ only if it has the corresponding (inverse) key k^{-1} (rules $\vdash_{enc}$ and $\vdash_{dec}$). Note that we are assuming that encryption is completely reliable. Indeed we do not allow any kind of cryptographic attack, e.g., the guessing of secret keys. This is the typical approach followed in the analysis of cryptographic protocols, which

$$(input)\frac{m \in Msg(c)}{c(x).E \xrightarrow{c(m)} E[m/x]} \qquad (output)\frac{m \in Msg(c)}{\overline{c}\,m.E \xrightarrow{\overline{c}\,m} E} \qquad (internal)\frac{}{\tau.E \xrightarrow{\tau} E}$$

$$(\|_1)\frac{E \xrightarrow{a} E'}{E \,\|\, E_1 \xrightarrow{a} E' \,\|\, E_1} \qquad (\|_2)\frac{E \xrightarrow{c(m)} E' \quad E_1 \xrightarrow{\overline{c}\,m} E_1'}{E \,\|\, E_1 \xrightarrow{\tau} E' \,\|\, E_1'} \qquad (+_1)\frac{E \xrightarrow{a} E'}{E + E_1 \xrightarrow{a} E'}$$

$$(=_1)\frac{m \neq m' \quad E_2 \xrightarrow{a} E_2'}{[m = m']E_1; E_2 \xrightarrow{a} E_2'} \qquad (=_2)\frac{m = m' \quad E_1 \xrightarrow{a} E_1'}{[m = m']E_1; E_2 \xrightarrow{a} E_1'} \qquad ([f])\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$$

$$(\backslash L)\frac{E \xrightarrow{a} E' \quad chan(a) \notin L}{E \backslash L \xrightarrow{a} E' \backslash L} \qquad (def)\frac{E[m_1/x_1, \ldots, m_n/x_n] \xrightarrow{a} E' \quad A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E}{A(m_1, \ldots, m_n) \xrightarrow{a} E'}$$

$$(\mathcal{D}_1)\frac{\langle m_1 \ldots m_r \rangle \vdash_{rule} m \qquad E_1[m/x] \xrightarrow{a} E_1'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_1'}$$

$$(\mathcal{D}_2)\frac{\nexists m : \langle m_1 \ldots m_r \rangle \vdash_{rule} m \qquad E_2 \xrightarrow{a} E_2'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_2'}$$

Fig. 2. Operational semantics.

permits to observe the attacks that can be carried out even if cryptography is completely reliable.

The formal behaviour of a CryptoSPA term is described by means of the LTS $< \mathcal{E}, Act, \{\xrightarrow{a}\}_{a \in A} >$, where $\xrightarrow{a}_{a \in A}$ is the least relation between CryptoSPA terms induced by axioms and inference rules of Figure 2 (where symmetric rules for $+_1$, $\|_1$ and $\|_2$ are omitted for the sake of readability). The operational semantics for a term E is the subpart of the CryptoSPA LTS reachable from the initial state E.

Example 1. We present a very simple example of a protocol where A sends a message m_A to B encrypted with a key k_{AB} shared between A and B. [3]

$$A(m,k) \stackrel{\text{def}}{=} [\langle m, k \rangle \vdash_{enc} x]\overline{c}\,x$$
$$B(k) \stackrel{\text{def}}{=} c(y).[\langle y, k \rangle \vdash_{dec} z]\overline{out}\,z$$
$$P \stackrel{\text{def}}{=} A(m_A, k_{AB}) \,\|\, B(k_{AB})$$

where $k_{AB}^{-1} = k_{AB}$, that models a symmetric encryption, and $Msg(c) = \{\{m\}_k \mid m \in M, k \in K\}$ that declares the "type" of messages sent over c. We want to analyze the execution of P with no intrusions, we thus consider $P \setminus \{c\}$, since the restriction guarantees that c is a completely secure channel. We obtain a system which can only execute action $\overline{out}\,m_A$ that represents the correct transmission of m_A from A to B. In particular, we have that $A(m_A, k_{AB}) \xrightarrow{\overline{c}\,\{m_A\}_{k_{AB}}} \underline{0}$

[3] For the sake of readability, we omit the termination $\underline{0}$ at the end of every agent specifications, e.g., we write a in place of $a.\underline{0}$. We also write $[m = m']E$ in place of $[m = m']E; \underline{0}$ and analogously for $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E; \underline{0}$.

and $B(k_{AB})$ can synchronize on that action by executing a $B(k_{AB}) \xrightarrow{c(\{m_A\}_{k_AB})}$ $[\langle\{m_A\}_{k_{AB}}, k_{AB}\rangle \vdash_{dec} z]\overline{out}\, z$. So

$$P \setminus \{c\} \xrightarrow{\tau} (\underline{0} \parallel [\langle\{m_A\}_{k_AB}, k_{AB}\rangle \vdash_{dec} z]\overline{out}\, z) \setminus \{c\} \xrightarrow{\overline{out}\, m_A} (\underline{0} \parallel \underline{0}) \setminus \{c\}$$

In the next section we will analyze the execution of this simple protocol in an hostile environment. ■

3 A General Schema for Security Properties

In this section we propose a general schema for the definition of security properties. We call it *Generalized* NDC (GNDC, for short), since it is a generalization of *Non Deducibility on Compositions* (NDC, for short) [9]. Indeed, in Section 4, we will show in details that NDC can be actually seen as a particular instance of GNDC. The main idea is the following: a system E is $GNDC^{\alpha}_{\triangleleft}$ *iff* for every environment X the composition of the system with X satisfies a specification $\alpha(E)$. Essentially, $GNDC^{\alpha}_{\triangleleft}$ guarantees that the property identified by α is satisfied (with respect to $\triangleleft$ relation) even when the system is composed with any possibly hostile environment. It determines a family of strong properties which are resistant to external attacks.

3.1 The GNDC Schema

In this section we formally define the $GNDC^{\alpha}_{\triangleleft}$ family of properties. First of all, we have to precisely specify what an "hostile environment" (or intruder) is. Basically, it is an agent which tries to attack a protocol by stealing and faking the information which is transmitted on the CryptoSPA *public* channels C. In principle, such an agent could be modeled as a generic process X which can communicate only through the channels belonging to C, i.e., $X \in \mathcal{E}_C$ where $\mathcal{E}_C \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid sort(E) \subseteq C\}$. However, in this way we obtain that X is a completely powerful attacker which is able to "guess" every secret information (e.g., cryptographic keys, nonces, private messages). Such an attacker can do whatever it wants over the protocol. Since it may know every secret, it can basically simulate every message exchanged in the protocol. Indeed, when we model cryptographic protocols, it is essential to specify that something is initially not known by the enemy. We clarify this (crucial) point through a simple example.

Example 2. Consider again the protocol P of Example 1. Since only A and B knows k_{AB}, this protocol should guarantee the authenticity of m_A even in the presence of an hostile environment. We assume that $c \in C$ is a public channel and we consider the following process:

$$X(m,k) \stackrel{\text{def}}{=} [\langle m,k\rangle \vdash_{enc} y]\overline{c}\, y$$

It belongs to $\mathcal{E}_C$ since $Sort(X(m,k)) = \{c\}$. Consider now $X(m_X, k_{AB})$ which is a process that knows k_{AB} and thus can send a faked message $\{m_X\}_{k_{AB}}$ to B.

In order to observe this we consider the following process "under attack" (note that we put X inside the scope of restriction):

$$(P \,\|\, X(m_X, k_{AB})) \setminus \{c\}$$

After one τ step, it can give as output $\overline{out}\, m_X$ which represent the fact that B as received m_X instead of m_A. This happens since $X(m_X, k_{AB})$ is in some sense "guessing" k_{AB}, but we would like to forbid it since, as mentioned above, we are interested in attacks that can be carried out even when cryptography is completely reliable. ∎

We solve this problem by imposing some constraints on the initial data that are known by the intruders. Given a process E, we call $ID(E)$ the set of messages that syntactically appear in E. Intuitively, this set contains all the messages that are initially known by E. Now, let $\phi_X \subseteq \mathcal{M}$ be the initial knowledge that we would like to give to the intruder, i.e., the public information such as the names of the entities and the public keys, plus some possible private data of the intruder (e.g., its private key or nonces). For a certain enemy X, we want that $ID(X)$ is consistent with ϕ_X. This can be obtained by simply requiring that all the messages in $ID(X)$ are deducible from ϕ_X.

The set $\mathcal{E}_C^{\phi_X}$ of processes which can communicate on a subset of C and have an initial knowledge bound by ϕ_X can be thus defined as follows:

$$\mathcal{E}_C^{\phi_X} = \{X \mid X \in \mathcal{E}_C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_X)\}$$

We consider as hostile processes only the ones belonging to this particular set. In the example above, if we require that k_{AB} is not deducible from ϕ_X (i.e., it is not public) we can easily see that the behavior of $X(m_X, k_{AB})$ cannot be simulated by any process in $\mathcal{E}_C^{\phi_X}$. As a matter of fact it can not execute $\overline{c}\,\{m_X\}_{k_{AB}}$, since we do not put k_{AB} in its initial knowledge. We will use $A \,\|_C\, B$ as a shortcut for $(A \,\|\, B)\backslash C$. The proposed family of security property is the following: [4]

Definition 2. *E is $GNDC_{\lhd}^{\alpha}$* iff

$$\forall X \in \mathcal{E}_C^{\phi_X} : E \underset{C}{\|} X \lhd \alpha(E)$$

where $\lhd \in \mathcal{E} \times \mathcal{E}$ is a relation between processes and $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ is a function between processes. ∎

We propose a sufficient criterion for a static characterization (i.e. not involving the universal predicate $\forall$) of $GNDC_{\lhd}^{\alpha}$ properties. We will say that $\lhd$ is a *pre-congruence* w.r.t. the operator $\|_C$ if it is a preorder and for every $E, F, F' \in \mathcal{E}$ if $F \lhd F'$ then $E \,\|_C\, F \lhd E \,\|_C\, F'$. Thus it is easy to prove the following:

Proposition 1. *If $\lhd$ is a pre-congruence w.r.t. $\|_C$ and if there exists a process $Top \in \mathcal{E}_C^{\phi_X}$ such that for every process $X \in \mathcal{E}_C^{\phi_X}$ we have $X \lhd Top$, then:*

[4] Indeed $GNDC$ depends on the set ϕ_X but we will not write it for the sake of readability.

$$E \in GNDC^{\alpha}_{\lhd} \qquad \textit{iff} \qquad E \underset{C}{\|} Top \lhd \alpha(E) \qquad \blacksquare$$

In particular, if the hypotheses of the proposition above hold then it is sufficient to check that $\alpha(E)$ is satisfied when E is composed to the most general (i.e., most powerful) environment Top. This is useful, since permits to make only one single check, in order to prove that a property holds whatever attacker we choose. We also have the following corollary for the congruence induced by $\lhd$:

Corollary 1. *Let $\lhd$ be a pre-congruence w.r.t. $\|_C$ and let $\equiv = \lhd \cap \lhd^{-1}$. If there exist two processes $Nil, Top \in \mathcal{E}_C^{\phi_X}$ such that for every process $X \in \mathcal{E}_C^{\phi_X}$ we have $Nil \lhd X \lhd Top$ then*

$$E \in GNDC^{\alpha}_{\equiv} \qquad \textit{iff} \qquad E \underset{C}{\|} Nil \equiv E \underset{C}{\|} Top \equiv \alpha(E) \qquad \blacksquare$$

3.2 Trace Equivalence

Most of the security properties that have been proposed for the analysis of security protocols are based on the simple notion of *traces*: two processes are equivalent if they exactly show the same execution sequences (called *traces*). In order to formally define traces, we need a transition relation which does not consider internal τ moves. This can be defined as follows:

Definition 3. *The expression $E \stackrel{\alpha}{\Longrightarrow} E'$ is a shorthand for $E(\stackrel{\tau}{\longrightarrow})^* E_1 \stackrel{\alpha}{\longrightarrow} E_2(\stackrel{\tau}{\longrightarrow})^* E'$, where $(\stackrel{\tau}{\longrightarrow})^*$ denotes a (possibly empty) sequence of τ labelled transitions. Let $\gamma = \alpha_1 \dots \alpha_n \in (Act \setminus \{\tau\})^*$ be a sequence of actions; then $E \stackrel{\gamma}{\Longrightarrow} E'$ if and only if there exist $E_1, E_2, \dots, E_{n-1} \in \mathcal{E}$ such that $E \stackrel{\alpha_1}{\Longrightarrow} E_1 \stackrel{\alpha_2}{\Longrightarrow} \cdots \stackrel{\alpha_{n-1}}{\Longrightarrow} E_{n-1} \stackrel{\alpha_n}{\Longrightarrow} E'$.* $\blacksquare$

We define *trace preorder* ($\leq_{trace}$) and *trace equivalence* ($\approx_{trace}$) as follows:

Definition 4. *For any $E \in \mathcal{E}$ the set $T(E)$ of* traces associated with E *is $T(E) = \{\gamma \in (Act \setminus \{\tau\})^* \mid \exists E' : E \stackrel{\gamma}{\Longrightarrow} E'\}$. F can execute all the traces of E (notation $E \leq_{trace} F$) iff $T(E) \subseteq T(F)$. E and F are* trace equivalent *(notation $E \approx_{trace} F$) iff $E \leq_{trace} F$ and $F \leq_{trace} E$, i.e., iff $T(E) = T(F)$.* $\blacksquare$

It is possible to prove that trace preorder $\leq_{trace}$ is a pre-congruence with respect to value-passing CCS operators.

Now we can provide the description of the most powerful intruder in the trace setting. It can be defined by using a family of processes $(Top^C_{trace})_\phi$ each representing the instance of the enemy with knowledge ϕ.

$$(Top^C_{trace})_\phi = \sum_{c \,\in\, C} c(x).(Top^C_{trace})_{\phi \cup \{x\}} + \sum_{\substack{c \,\in\, C \\ m \,\in\, \mathcal{D}(\phi) \cap Msg(c)}} \overline{c}\, m.(Top^C_{trace})_\phi$$

The following holds:

Proposition 2. *If $X \in \mathcal{E}_C^{\phi_X}$ then $X \leq_{trace} (Top^C_{trace})_{\phi_X}$.* $\blacksquare$

So, we have proved that there exists a top of the set $\mathcal{E}_C^{\phi_X}$ with respect to $\leq_{trace}$ and it is indeed $(Top^C_{trace})_{\phi_X}$. These result, together with the fact that $\leq_{trace}$ is a pre-congruence with respect to $\|_C$ allow us to apply Proposition 1 obtaining the following result for the family of $\leq_{trace}$-based $GNDC^{\alpha}_{\leq_{trace}}$ properties:

Corollary 2. *For every function* $\alpha : \mathcal{E} \to \mathcal{E}$

$$E \in GNDC^{\alpha}_{\leq_{trace}} \qquad \textit{iff} \qquad E \underset{C}{\|} (Top^C_{trace})_{\phi_X} \leq_{trace} \alpha(E) \qquad \blacksquare$$

Note that this corollary holds for every possible α. So, every property which is based on trace pre-congruence can be checked statically. We show how we can directly apply this result on our simple running example.

Example 3. Consider again protocol P of Example 1. We would like to check that no intruder is able to fake message m_A if it does not know the shared key k_{AB}. We do this by checking that $P \in GNDC^{\overline{out}\, m_A}_{\leq_{trace}}$. Indeed, if this holds then even in the presence of any enemy X, the process B only receives message m_A and no fake is possible. We assume that $c \in C$, $\phi_X = \emptyset$ and, as before, $Msg(c) = \{\{m\}_k \mid m \in M, k \in K\}$. By corollary 2 we have that $P \in GNDC^{\overline{out}\, m_A}_{\leq_{trace}}$ if and only if $P' \stackrel{\text{def}}{=} P \|_C (Top^C_{trace})_\emptyset \leq_{trace} \overline{out}\, m_A$. Since every action a, executed by the enemy and such that $chan(a) \neq c$ will never synchronize with P, then it is easy to see that $P' \approx_{trace} P \|_C (Top^{\{c\}}_{trace})_\emptyset$. Now, if A and B communicate together we have $P' \stackrel{\overline{out}\, m_A}{\Longrightarrow} (\underline{0} \| \underline{0} \| (Top^{\{c\}}_{trace})_\emptyset) \setminus \{c\}$ which can do nothing else. Otherwise, A and B could communicate with the enemy. Note that $(Top^{\{c\}}_{trace})_\emptyset$ can only read from channel c since it has no knowledge. So, the only possible move is the one where it intercepts the message from A:

$$P' \xrightarrow{\tau} (\underline{0} \| B(k_{AB}) \| (Top^{\{c\}}_{trace})_{\{m_A\}_{k_{AB}}}) \setminus \{c\}$$

It is easy to see that $(Top^{\{c\}}_{trace})_{(\{m_A\}_{k_{AB}})}$ can only either read again from c or send $\{m_A\}_{k_{AB}}$ on c. Indeed, it cannot decrypt the message since it does not know the key and it cannot send other "kind" of messages on c such as pairs composed by two instances of $\{m_A\}_{k_{AB}}$ because of how is defined $Msg(c)$. Since none is sending messages on c we have only one possible execution:

$$(\underline{0} \| B(k_{AB}) \| (Top^{\{c\}}_{trace})_{\{m_A\}_{k_{AB}}}) \setminus \{c\} \stackrel{\overline{out}\, m_A}{\Longrightarrow} (\underline{0} \| \underline{0} \| (Top^{\{c\}}_{trace})_{\{m_A\}_{k_{AB}}}) \setminus \{c\}$$

We conclude that the only trace executable by P' is $\overline{out}\, m_A$, thus $P' \leq_{trace} \overline{out}\, m_A$ and $P \in GNDC^{\overline{out}\, m_A}_{\leq_{trace}}$. It is analogously possible to prove that $P \in GNDC^{\overline{out}\, m_A}_{\leq_{trace}}$ even when ϕ_X is not empty (the enemy could know some private messages or keys) and we simply require that $k_{AB} \notin \mathcal{D}(\phi_X)$. $\blacksquare$

4 Some Examples of Security Properties

In this section we want to show the generality of $GNDC^{\alpha}_{\lhd}$ properties. In particular we show that a number of existing formal definitions of security properties can be redefined as $GNDC^{\alpha}_{\lhd}$ ones, with particular instantiations of α and $\lhd$.

4.1 Non Deducibility on Compositions

Non Deducibility on Compositions (NDC, for short) [9, 10] has been proposed as a generalization of the classical idea of *Non-Interference* [11] to non-deterministic systems. Non-Interference tries to capture whether a certain group of processes G is able to "interfere" in some way with another group G', i.e., if what is done by processes in G has some effect on the execution of processes in G'. Sometimes Non-Interference properties are also called *Information Flow* properties, since an interference of G with G' can be seen as a information flow from the first group to the second one. A classical application of these properties is *multilevel security* [3] where H represent a set of "classified" (high level) processes that should be forbidden to send any data to L (low level), i.e., to interfere with L. In [8] NDC has also been applied to the verification of security protocols.

Since $GNDC^{\alpha}_{\triangleleft}$ is a generalization of NDC it can be instantiated in order to obtain NDC and also the *bisimulation* based NDC, called BNDC. We first redefine in our extended language the original definitions: we consider C as the set of channels that classified processes H use when trying to interfere with the processes L. Thus, H corresponds to $\mathcal{E}_C^{\phi_X}$ and NDC can be defined as follows:

Definition 5. *E is NDC if and only if $\forall \Pi \in \mathcal{E}_C^{\phi_X}$, $E \setminus C \approx_{trace} E \,\|_C\, \Pi$.* ■

where the only difference with respect to the definition given in the original model is that the knowledge of processes $\Pi \in H$ is bounded by ϕ_X. In the extended model, this is required to guarantee reliable encryption. NDC requires that high level processes $\mathcal{E}_C^{\phi_X}$ are not able to change the low level behaviour of the system represented by $E \setminus C$. As a matter of fact $E \setminus C$ is the system where no high level activity is allowed. If it is equivalent to $E \,\|_C\, \Pi$ this clearly means that Π is not able to modify in any way the execution of E.

We can obtain a bisimulation based NDC by simply substituting $\approx_{trace}$ with $\approx_{bisim}$. We do not define bisimulation here, since we will not use it directly, but we only mention that it is a strong observational equivalence which requires that two bisimilar processes are able to simulate each other step by step (see, e.g., [20] for more details).

Definition 6. *E is BNDC if and only if $\forall \Pi \in \mathcal{E}_C^{\phi_X}$, $E \setminus C \approx_{bisim} E \,\|_C\, \Pi$.* ■

Note that NDC and BNDC correspond to $GNDC^{E\setminus C}_{\approx_{trace}}$ and $GNDC^{E\setminus C}_{\approx_{bisim}}$, respectively. For NDC it is also possible to apply Corollary 1 obtaining an interesting static characterization.

Proposition 3. *E is NDC iff $E \,\|_C (Top^C_{trace})_{\phi_X} \approx_{trace} E \setminus C$.* ■

This result is the analogous of the one in [9]. Note that here we have found it as a particular case of the more general result of Corollary 1.

For BNDC we cannot give an analogous static characterization. Indeed, to the best of our knowledge, the only preorder whose kernel is the weak bisimulation, is the weak bisimulation itself. Thus, in this case, we cannot find suitable processes *Nil* and *Top*. As a matter of fact, such processes would result to be bisimilar.

It is worthwhile noticing that there are several similarities with a related problem in temporal logic, namely *module checking*. Given a finite system, which is able to interact with its environment, the *module checking* problem is the task of verifying that for every environment the induced behaviour of the system satisfies a certain temporal logic formula. Interestingly, if the formula expresses only safety properties (i.e. trace based properties) then the problem is reduced to check the system in composition with the environment which enables every system transition (the most general one!). But if the formula expresses also liveness properties then problem becomes very difficult and it is no more sufficient to consider the most general environment (see [19] for a deeper discussion).

4.2 The Agreement Property

In this section we show that also the approach proposed in [15] for the analysis of authentication properties, inside the framework of CSP [12] process algebra, can be rephrased in terms of our specification schema. The basic idea of the *Agreement* property is the following:

> "A protocol guarantees to an initiator A *Agreement* with a responder B on a set of data items ds if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of A corresponds to a unique run of B".

What is technically done in the *Agreement* property is to have for each party an action representing the running of the protocol and another one representing the completion of it. For example, consider an action $commit\_res(B, A, d)$ representing a correct termination of B as a responder that is convinced to communicate with A and agrees on data in d. Moreover we have an action $running\_ini(A, B, d)$ that represents the fact that A is running the protocol as initiator, apparently with B and with data d. If we have these two actions specified in the protocol, the *Agreement* property requires that when B executes $commit\_res(B, A, d)$ then A has previously executed $running\_ini(A, B, d)$. This means that every time B completes the protocol with A convinced that the relevant data are the ones represented by d, then A must have been running the protocol with B using exactly the data in d.

As done in [15], we assume that the actions representing the running and the commit are correctly specified in the protocol. We can see them as output actions over two particular channels $running\_ini$ and $commit\_res$. For simplicity, we only analyze the case where A is the initiator and B is the responder, and the set ds of variables is composed only by d which can assume values in a set D. However, the specification can be easily extended in order to cover all the cases studied in [15]. Function $\alpha(E)$ can be defined as follows:

$$\begin{aligned} E' &= Top_{trace}^{Sort(E)\setminus\{running\_ini, commit\_res\}} \\ E''(x,y) &= \sum_{d\in D} \overline{running\_ini}\,(x,y,d) \;\;.\;\; \overline{commit\_res}\,(y,x,d) \\ \alpha_{Agree}(E) &= E' \parallel E''(A,B) \end{aligned}$$

Given E, $\alpha(E)$ represents the most general system which satisfies the agreement property and has the same sort as E. As a matter of fact in $\alpha(E)$ the action $\overline{running\_ini}\,(A,B,d)$ always precedes $\overline{commit\_res}\,(B,A,d)$ for every datum d, and every combination of the other actions of E can be executed. In order to analyze more than one session, it is sufficient to consider an extended α which has several processes $E''(A,B)$ in parallel. For example, for n sessions we can consider the following:

$$\alpha_{Agree}(E) = E' \parallel \underbrace{E''(A,B) \parallel \ldots \parallel E''(A,B)}_{n}$$

We want that even in the presence of an hostile process, E does not execute traces that are not in $\alpha(E)$,i.e., we require that $E \parallel_C X \leq_{trace} \alpha(E)$. So we can give the following definition:

Definition 7. *E satisfies Agreement iff E is* $GNDC_{\leq_{trace}}^{\alpha_{Agree}(E)}$. ■

Note that in [15] it is only required that *Agreement* holds when the system is composed with a particular intruder, which turns out to be equivalent to the most general one. In the following we exploit Proposition 1 in order to formally prove that such a (static) requirement is indeed sufficient (and necessary) to guarantee our *GNDC*-based version of *Agreement.* As a matter of fact, by Corollary 2 we immediately have the following result:

Proposition 4. *E satisfies Agreement iff* $E \parallel_C (Top_{trace}^C)_{\phi_X} \leq_{trace} \alpha_{Agree}(E)$. ■

In [15], other versions of *Agreement* are defined. We can rephrase all of them in our model by simply changing the α function. [5]

4.3 Message-Oriented Authentication

Now, we consider the message-based approach to authentication defined in [24, 26] using the CSP language. The idea is to observe when a set of messages T authenticates another set of messages R. Informally, T authenticates R if the occurrence of some element of T implies the occurrence of some element of R (it is required that T and R are disjoint). When a system P satisfies this property we say that P satisfies T **authenticates** R.

In [24] the net is represented by a process $Medium$ which acts like a router by receiving and forwarding to the correct process the messages. In CSP, it is

[5] Indeed, *recentness* cannot be immediately rephrased in our CCS-based model, because of the difference in handling communication with respect to CSP. This could be overcome by extending our language with time as done in [19]. This is only related to the differences in the model, and is not caused by a weakness of our schema.

possible to observe the communication between the processes and the medium since they are not "internalized" as in CCS. However, we can simulate this by assuming that the *Medium* echoes every routing of messages through particular output actions on two reserved channels *trans* and *rec*. For example $\overline{trans}\,(A, B, m)$ corresponds to the sending of message m from A to B and, symmetrically, $\overline{rec}\,(B, A, m)$ represents the reception of it. In this way we can observe communication as done in CSP. Sets T and R range over these reserved actions. We can now define the $\alpha_{auth_R^T}(E)$ function as follows:

$$\begin{aligned} \alpha_{auth_R^T}(E) &= E' \\ E' &= \Big(\sum_{\substack{a \,\in\, Act \\ a \,\notin\, R \cup T}} a.E'\Big) + \sum_{\substack{a \,\in\, Act \\ a \,\in\, R}} a.E'' \\ E'' &= Top_{trace}^{Sort(E)} \end{aligned}$$

Process $\alpha_{auth_R^T}(E)$ can execute actions in T only after it has executed some actions in R. Indeed we note that it moves to E'', where it can execute also actions in T, only after it performs at least one action in R. This is exactly what we require by our system E and $\alpha(E)$ is indeed the most general system (with the same sort as E) satisfying T **authenticates** R. So we can give the following definition:

Definition 8. *E satisfies T* **authenticates** *R iff E is $GNDC_{\leq_{trace}}^{\alpha_{auth_R^T}(E)}$.* ■

As in the section above, we can prove that the approach followed in [24], where it is considered only the most powerful intruder, guarantees that the property holds in the presence of whatever hostile process. By Corollary 2 we obtain that:

Proposition 5. *E satisfies T* **authenticates** *R iff $E \,\|_C (Top_{trace}^{C})_{\phi_X} \leq_{trace} \alpha_{auth_R^T}(E)$.* ■

4.4 Non-repudiation

In this section, we show that also non-repudiation properties can be formulated within the *GNDC* schema. Non repudiation protocols have the aim of producing evidence about the execution of services, among parties that do not trust each other (see [27, 28]).

In [25] Schneider shows how to apply verification methods based on *CSP* process algebra to the analysis of a (fair) non repudiation protocol proposed in [27]. Among the non repudiation properties studied in [25, 28], we briefly recall:

– *Non Repudiation of Origin (NRO)* is intended to protect the receiver from the false denial of another party to have sent a message.
– *Non Repudiation of Receipt (NRR)* is intended to protect the sender form the false denial of another party to have received a message.

Roughly speaking, the analysis performed by Schneider is similar to his message based authentication (see section above). As an example, consider NRO verification: if the receiver is able to produce an evidence of the sending of a certain message m, then, actually, m should have been sent. In other words, such an evidence should "authenticate" m (in the sense of message-based authentication).

Non-repudiation protocols are not concerned with communication among two or more parties in an hostile environment. So, at a first glance, the $GNDC$ schema seems to be not applicable. However, the parties do not trust each other, and in particular, one of them could try to act maliciously in order to obtain some advantage. We will show that this malicious party may be considered as the hostile environment in a $GNDC$ schema.

In the verification of NRO (NRR) we assume that a Judge should be able to establish that a certain message has been sent (received) if he obtains some evidence of it from the receiver (sender). This verification should be carried out by simply assuming that both the sender and the receiver have not sent on the net some information which could invalidate the evidence. In particular, the Judge cannot assume that they have followed the protocol. For this reason, Schneider models both the sender and the receiver similarly to the most general intruder. In order to apply the $GNDC$ schema, we consider a weaker (but still reasonable) notion of NRO; in particular, we require that if the receiver B, after following the protocol, is able to give evidence of origin, then the sender A has actually sent that message. We call this NRO with honest receiver NRO_{hr}. It can be simply encoded as $GNDC$ schema by considering a process E_B where we only have the receiver B plus the possible communication medium (but we do not specify A):

$$\forall X_A \in \mathcal{E}_C^{\phi_A} \quad X_A \underset{C}{\|} E_B \leq_{trace} \alpha^{wnro}(E_B) \tag{2}$$

Now, if R is the set of all the actions where m is sent as message and ev_of_or is the action which signals that B has evidence of origin of m, then $\alpha^{wnro}(E_B)$ can be defined as $\alpha_{auth_R^{\{ev\_of\_or\}}}(E_B)$. [6] Analogously, we define non-repudiation of origin with honest sender, i.e., NRO_{hs}. This property can be encoded in the $GNDC$ schema (2) by simply considering process E_A instead of E_B and by quantifying over processes which have the initial knowledge ϕ_B. Symmetrically to E_B, in E_A only A and the possible medium are given, while B is left unspecified.

We can now define *weak-NRO* as the intersection of NRO_{hr} and NRO_{hs}, i.e., $E_A, E_B \in GNDC^{\alpha^{wnro}}_{\leq_{trace}}$. An analogous definition may be given for *weak-NRR*, even though the situation is slightly more complicated since, in this case, also liveness properties should be considered. We do not address this issue in details, since we prefer to focus our attention to another property which is also based on liveness. The property is *fairness* [28]:

[6] Indeed, there is a slight limitation due to the necessity of preventing the communication of information that could invalidate the proof of evidence. This can be treated in our model as done in [25].

– *Fairness*: At no point in the protocol run does either of participants have an advantage. In other words no one of the party can get his own evidence and avoid the other to get his corresponding evidence.

As observed in [25], this property cannot be defined as a safety property (i.e. nothing bad happens). Indeed we have to prove that whenever one of the two participants obtains his own evidence, then the other must be in the position to get his own evidence too. This can be seen as a liveness property (i.e. something good happens). For this reason, in the analysis of this property it is used the *failure* model instead of the trace one. As a matter of fact, failure equivalence is actually able to observe potential deadlocks in the executions, and so it permits to see if something can be executed or not (i.e., if an evidence can be obtained or not). The verification technique for the fairness property proposed in [25] directly fits in the $GNDC$ schema. Indeed, it is reasonable to assume that an agent can require fairness from the other only in the case he behaves correctly, i.e., if it follows the protocol. For example, the *fairness* for the sender A of receiving evidence of message receipt can be defined as in (2) with a suitable relation $\leq_{failure}$ which takes into account failures, and a function α^{fair} which models the fact that after the receiver gets evidence of the origin (ev_of_or) then the sender have the possibility to obtain his own evidence of receipt (ev_of_re). This is modeled in α^{fair} by making the action ev_of_re always executable after that ev_of_or has been engaged.

4.5 Authentication in the Spi-Calculus

In [2, 1] an interesting notion of authentication is proposed. The basic idea is the following: consider a protocol $P(M)$, which tries to transmit message M from one party (say A) to another one (say B). The authentication of the message M is checked by verifying if $P(M)$ is equivalent to a specification $P_{spec}(M)$ where M is always delivered correctly. In $P_{spec}(M)$ the receiver B always knows M and whatever happens on the communication channel, B will continue its execution exactly as it had received the correct message M. In other words, $P_{spec}(M)$ represents the situation where M is always received and no enemy is able to substitute it with a different message. If $P(M)$ is equivalent to $P_{spec}(M)$ then also $P(M)$ is clearly able to avoid any possible attack. The language used in [2, 1] is the spi-calculus. Moreover the may-testing equivalence (see [5]) is used in order to check that $P(M)$ is equivalent to $P_{spec}(M)$ with respect to any possible interaction with the (hostile) environment. The definition of authentication in the spi-caluclus is given as follows:

Definition 9. *$P(M)$ guarantees authentication if and only if for all M we have that $P(M)$ is may-testing equivalent to $P_{spec}(M)$* ■

It seems reasonable to rephrase this property in our model as follows:

Definition 10. *Let $P(M)$ be a protocol where the parties communicates over the set C of channels. $P(M)$ guarantees authentication iff for all M we have that $P(M) \in NDC$.* ■

NDC is the one defined in Section 4.1. In particular, $P(M) \in NDC$ requires that $P(M)$ composed with whatever enemy X is (trace) equivalent to $P(M) \setminus C$. Since C represents the set of channels over which the parties communicate, then $P(M) \setminus C$ corresponds to our secure specification $P_{spec}(M)$ where no enemy is able to modify the execution of the protocol. In the spi-calculus the system is checked against all the possible interactions with the (hostile) environment through the use of the may-testing equivalence. Here we do it explicitly through the quantification over X which is the base of the $GNDC$ schema.

Indeed, Definitions 9 and 10 are based on two quite different models. As a matter of fact, in the spi-calculus it is possible to send a channel as a message, thus giving the possibility of creating dynamically a new secure channel. However, we are quite confident that NDC, as defined in our model, is strong enough to detect the attacks shown in [2]. Note also that NDC can be statically characterized, thus permitting a simplification of the verification task. We are presently working on a formal comparison of these two definitions in the spi-calculus setting. It is worthwhile noticing that the knowledge of the enemy is handled differently in our model and in the spi-calculus. In CryptoSPA, we assume that the initial knowledge of the enemy is represented by ϕ_X. In the spi-calculus we find a complementary modeling of this since there is no specification of the initial knowledge on the enemy, and the restriction operator is exploited in order to guarantee that some information is kept secret from anyone other than the parties of the protocol. Indeed, the restriction operator of the spi-calculus can be seen as a generator of fresh names that cannot be guessed by any (whatever powerful!) enemy. Note that, given a protocol P in the spi-calculus, we can obtain ϕ_X by simply requiring that the restricted names in P are not in ϕ_X (in this sense we have a complementary modeling of knowledge).

4.6 Comparison

In this section we show one of the advantages in having a uniform treatment of security properties, namely the possibility of studying the relationships among them in a fairly simple way. First, we show that NDC may be seen as a sufficient condition for every property which is based on trace-preorder. This result is interesting since it relates NDC (first proposed for modeling information flow security) to properties which has been proposed for completely different purposes, e.g., authentication. The result holds for what we will call *good candidates* for a function α, i.e., processes E such that $E \setminus C \leq_{trace} \alpha(E)$. This condition is quite reasonable since we certainly want that at least the protocol under no attacks (i.e., $E \setminus C$) "satisfies" $\alpha(E)$.

Proposition 6. *Let $\alpha(E)$ be a function between processes and let E be a good candidate for α, i.e., $E \setminus C \leq_{trace} \alpha(E)$. Then, E is $GNDC^{E \setminus C}_{\approx_{trace}}$ implies that E is $GNDC^{\alpha(E)}_{\leq_{trace}}$.* ■

Note that if a function α does not have good candidates then it represents an empty property (no process satisfies it). Note also that every process E is a good candidate for $GNDC^{E \setminus C}_{\approx_{trace}}$.

The result above shows that $GNDC^{E\backslash C}_{\approx_{trace}}$ is stronger that $GNDC^{\alpha_{Agree}(E)}_{\leq_{trace}}$ and $GNDC^{\alpha_{auth}(E)}_{\leq_{trace}}$ for their respective good candidates. This is quite intuitive since $GNDC^{E\backslash C}_{\approx_{trace}}$ basically requires that the protocol behaviour is completely preserved even under attacks. So if E satisfies a certain property under no attacks (i.e., it is a food candidate), then $GNDC^{E\backslash C}_{\approx_{trace}}$ will preserve such a property also under every possible attack.

In general, we observe that if $\lhd \subseteq \lhd'$ then $GNDC^{\alpha}_{\lhd} \subseteq GNDC^{\alpha}_{\lhd'}$, furthermore if for all $E \in \mathcal{E}$ we have $\alpha(E) \lhd \alpha'(E)$ then $GNDC^{\alpha}_{\lhd} \subseteq GNDC^{\alpha'}_{\lhd}$.

Indeed, both *Agreement* and *message-authentication* are based on traces and so the relations between them can be easily derived by comparing their relative α. Here, we give an example of comparison of such properties in the simple case of a single run between a sender A and a receiver B. Moreover we also assume that we have no variables to agree on. We consider $T = \{commit(B, A)\}$ and $R = \{running(A, B)\}$. It is easy to prove that, for all $E \in \mathcal{E}$:

$$\alpha_{agree}(E) \leq_{trace} \alpha_{auth^{\{commit(B,A)\}}_{\{running(A,B)\}}}(E).$$

We also give a simple example of how it is possible to exploit property comparisons in the verification of protocols. We consider a slight variant of example 1 where messages $running(A, B)$ and $commit(B, A)$ are suitably inserted.

$$\begin{aligned} A(m,k) &\stackrel{\text{def}}{=} [\langle m,k\rangle \vdash_{enc} x]running(A,B).\overline{c}\,x \\ B(k) &\stackrel{\text{def}}{=} c(y).[\langle y,k\rangle \vdash_{dec} z]commit(B,A) \\ P &\stackrel{\text{def}}{=} A(m_A,k_{AB}) \,\|\, B(k_{AB}) \end{aligned}$$

It is easy to see that the composed system P is a good candidate for α_{agree}, and so also for α_{auth}, when we consider $C = \{c\}$. Moreover by using arguments similar to the ones of Example 3, we can prove that $P \in GNDC^{E\backslash C}_{\approx_{trace}}$. Thus by proposition 6 we get that $P \in GNDC^{\alpha_{agree}}_{\leq_{trace}}$ and $P \in GNDC^{\alpha_{auth}}_{\leq_{trace}}$, where $\alpha_{auth} = \alpha_{auth^{\{commit(B,A)\}}_{\{running(A,B)\}}}(E)$.

5 Conclusions

In this paper we have proposed a uniform method for defining computer security properties. In doing so, we have tried to exploit some underlying ideas of existing proposals rather than giving a completely new approach. Actually, we did not try to obtain a universal definition for all possible security properties; our aim was indeed to find a quite flexible and useful schema that could help in reasoning about different properties. Indeed, we have shown the flexibility of our proposal by rephrasing on it a number of existing definitions, some of which have completely different aims (e.g., Non-Interference, authentication and non-repudiation).

We think that a uniform schema for the definition of security properties could have several advantages. First, it allows to study the relationships among different security properties. For example, we have shown that NDC may be seen as the strongest property definable for a certain protocol since it requires that the action of any intruder does not modify in any way the observable behaviour of the protocol. Other formal relationships between security properties can be easily established by simply reasoning on α and $\lhd$. As a future work we will try to give a general taxonomy of security properties, which could extend and possibly integrate the one for information flow given in [9] and the one for authentication protocol properties studied in [15]. We are presently studying if also other security properties can be conveniently defined in our general framework.

By using a unique model and a unique schema it is possible to develop more general theories which could then be applied to a number of definitions by simply instantiating them. For example, we have shown that the static characterization result, which permits to check a property only against the most general intruder, can be easily applied to all the trace-semantics based properties. It would be interesting to extend some proof techniques for the analysis of authentication protocols (e.g., the ones developed in [14, 26, 7]), in order to deal with the GNDC schema. In this way they could also be applied for the analysis of other GNDC security properties.

Another interesting point is that the *GNDC* schema itself suggests new analysis techniques. Indeed, the analysis of *GNDC*-like properties may be seen as example of *module checking*, i.e. model checking of systems which have to interact with arbitrary environments. Thus compositional analysis concepts exploited in [19] for the analysis of such problems may be recasted also for the analysis of security problems. This approach has been successfully followed in [17, 18] where an automated methodology for the analysis of GNDC like properties over protocols with a finite behaviour has been developed. This methodology does not require the specification of a particular intruder and may be thus adopted also when the hypothesis of proposition 1 does not hold.

Acknowledgements: We would like to thank Prof. Roberto Gorrieri for the interesting hints and discussions on the topic of this paper. Furthermore, we wish to thank the anonymous referees for their helpful comments and suggestions.

References

[1] M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proceedings of CONCUR'97*, pages 59–73. LNCS 1243, 1997.

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

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

[4] C. Bodei, P. Degano, R. Focardi, and C. Priami. Authentication via localized names. In *Proceedings of CSFW'99*, pages 98–110. IEEE press, 1999.

[5] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[6] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.

[7] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proceedings of CSFW'99*, pages 203–212. IEEE press, 1999.

[8] R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

[9] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[10] R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

[11] J. A. Goguen and J. Meseguer. Security policy and security models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Press, 1982.

[12] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[13] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[14] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS'96*, pages 146–166. LNCS 1055, 1996.

[15] G. Lowe. A hierarchy of authentication specification. In *Proceedings of the 10th Computer Security Foundation Workshop*, pages 31–43. IEEE press, 1997.

[16] W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In *Proc. of DIMACS Workshop on Design and Formal Verification of Security Protocols.* Rutgers University, Sep. 1997.

[17] F. Martinelli. Languages for description and analysis of authentication protocols. In *Proceedings of ICTCS'98*, pages 304–315. World Scientific, 1998.

[18] F. Martinelli. Partial model checking and theorem proving for ensuring security properties. In *Proceedings of CSFW'98*, pages 44–52. IEEE press, 1998.

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

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

[21] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-State Analysis of SSL 3.0. In *7th USENIX Security Symposium*, 1998.

[22] P. Y. A. Ryan and S. Schneider. Process algebra and non-interference. In *Proceedings of CSFW'99*, pages 214–227. IEEE press, 1999.

[23] A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall, 1997.

[24] S. Schneider. Security Properties and CSP. In *Proceedings of the 1996 Symposium on Security and Privacy*, pages 174–187. IEEE Press, 1996.

[25] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of CSFW'98*, pages 54–65. IEEE Press, 1998.

[26] S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), September 1998.

[27] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proc. of Symposium in Research in Security and Privacy*, pages 55–61. IEEE Press, 1996.

[28] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *International Refinement Workshop and Formal Methods Pacific*, 1998.