A Uniform Approach for the Analysis of Cryptographic Protocols *

(Extended Abstract)

Riccardo Focardi[1] and Fabio Martinelli[2]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia
via Torino 155, I – 30173 Mestre (Ve), Italy.
e-mail: focardi@dsi.unive.it

[2] Dipartimento di Informatica, Università di Pisa
Corso Italia 40, I – 56125, Pisa, Italy.
e-mail: 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 whatever hostile environment. This determines a family of strong properties which are resistant to every external attacks, but are quite impractical to check. For this reason, we find some general conditions that make it sufficient to check a property only against a "most powerful" intruder. In [11], we show that general results of our theory can be easily applied to a number of existing security properties that can be rephrased in our setting. As an example, we apply our definition to a simplified version of the Wide Mouthed Frog protocol.

Achievements: In this paper we present a general schema for the definition of typical cryptographic protocol properties, e.g., authentication, secrecy, non-repudiation. A number of existing properties (e.g., [9, 10, 15, 22, 2, 1, 21]) can be rephrased in our schema. All of them are based on models that consider cryptography as a secure underlying primitive. Indeed, also in such models interesting attacks to security can be detected, i.e., attacks on protocols that can be done without even breaking cryptography. We show that general properties of our schema can be easily instantiated to particular security definitions. We hope that our proposal could be useful to compare and better understand the various security properties defined in literature.

1 Introduction

In the last years, a number of new flaws in cryptographic protocols have been discovered (see, e.g., [4] for a survey). Most of them are due to some weakness in the message sequence of the protocols and are not obtained by breaking the cryptographic algorithms used by the protocols. This kind of attacks has focussed the attention of many researchers on the development of models which allow to verify if a certain protocol is flawed when assuming that the underlying cryptographic primitives are completely secure (see, e.g., [2, 1, 7–10, 12, 13, 15–17, 19, 22]).

Typically, when we require that one of these security properties is satisfied we would like to be guaranteed that it 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.

* Work partially supported by MURST Progetto "Tecniche Formali per la Specifica, l'Analisi, la Verifica e la Trasformazione dei Sistemi Software". An extended version of this paper can be found in proceedings of *World Congress on Formal Methods* (FM99).

In this work, we propose a general schema for the definition of these security properties which is actually based on the idea of checking a system against all the possible hostile processes. Such an idea was already studied in [8, 9, 16] for a particular property called *Non Deductibility on Compositions* (NDC, for short). Our general schema is indeed 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 whatever (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 whether process E shows a correct behaviour or not, even in the presence of X. The underlying model of communication guarantees that X cannot decrypt a message if it does not have the corresponding key. Cryptography is thus considered as a secure black-box.

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 [5, 14, 20]). 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 does exist, and, if it exists, 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" the property to the case where only the most general intruder is considered. When this is possible, we indeed obtain a formal proof that the "most powerful" intruder approach is correct.

It is possible to see that the approach proposed here is general by rephrasing a number of existing properties as particular instances of (1). For example, in [11] we define *NDC*, *BNDC* [9, 10], *Agreement* [15], *authentication* as proposed in [22] and [2, 1] and *non-repudiation* [21] by choosing particular instances of $\lhd$ and $\alpha(E)$. In particular, *authentication* for many of them, the $\lhd$ relation corresponds to *trace preorder* which satisfies the conditions for the static characterization, thus permitting to "safely" apply the most general intruder approach for the analysis of such properties (as we will show in this paper).

Our general scheme permits also to study relationships among different security properties. In [11] we show that 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* [12] in non-deterministic systems while *Agreement* has been proposed for the analysis of entity authentication in protocols. This gives some interesting results, e.g., the fact that 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 an extension of value-passing CCS [18] which has been provided with two auxiliary operators for the modelling of cryptography. The details of the semantics are given in Appendix A. Then, in Section 3, we define our general schema and we give some general results on it. Section4 is devoted to an example that shows how our property is used in practice. Finally, Section 5 contains some concluding remarks.

2 The Model

In this section we briefly present the language we will use for the specification of security properties and protocols. It is basically an extension of value-passing CCS [18] which permits to handle cryptography. In particular, we give to processes the possibility of encrypting and decrypting messages and of constructing "complex" messages by composing together simple ones. We also add a new operator of *knowledge restriction* which can be used to avoid the guessing of secret keys and messages. 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.

2.1 The CSPA language

As mentioned above, the language we will use for specifying protocols is and an extension of CCS and in particular of the *Security Process Algebra* (SPA for short) proposed in [10] for the description and the analysis of multi-level systems. This extension borrows some concepts from the language proposed in [16] for the analysis of authentication protocols. We call the calculus *Cryptographic Security Process Algebra* (CSPA for short).

CSPA syntax is based on the following elements:

- A set $I = \{a, b, \dots\}$ of *input* channels, a set $O = \{\bar{a}, \bar{b}, \dots\}$ of *output* channels;
- 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 $Msg(c) : \mathcal{L} \longrightarrow \mathcal{U}$ 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, ranged over by a, where τ is the internal (invisible) action and $c(m), \overline{c}\,m$ represent the input and the output of message m over the channel c, respectively. 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 $a = \tau$.
- A set $Const$ of constants, ranged over by A.

The syntax of CSPA agents is defined as follows:

$$E ::= \underline{0} \;\big|\; c(x).E \;\big|\; \overline{c}\,m.E \;\big|\; \tau.E \;\big|\; E+E \;\big|\; E \,\|\, E \;\big|\; E \setminus L \;\big|\; E[f] \;\big|$$
$$\big|\; A(v_1, \ldots, v_n) \;\big|\; [m = m']E; E \;\big|\; (E)_\phi \;\big|\; [\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E; E$$

where $x, m, n, k, m_1, \ldots, m_r$ are variables that will be instantiated to messages, $v_1 \ldots v_n$ are messages, ϕ is a set of messages and L is a set of input channels. Both the $c(x).E$ operator and the $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E; E'$ one bind the variable x in E.

It is also necessary to define constants as follows: $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E$ where E is a CSPA 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 where 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 $f(c)$; $A(v_1, \ldots, v_n)$ behaves like the respective definition where all the variables $x_1, \ldots, x_n$ are substituted with messages $v_1, \ldots, v_n$; 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 two additional operators that have been introduced in order to model cryptography. Intuitively $(E)_\phi$ behaves like E which only "knows" the messages in ϕ; this means that E cannot give as output messages that are not deducible from ϕ; for example $(E)_\phi$ is not able to give as output the decryption of a message if it does not know the corresponding inverse key, i.e., if that (inverse) key is not deducible from ϕ; as expected $(E)_\phi$ increases its knowledge to $\phi \cup \{m\}$ every time it receives a message m through an input action; 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 $\langle m_1 \ldots m_r \rangle$ through 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 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 CSPA terms, and we define $sort(E)$ to be the set of all the channels syntactically occurring in the term E.

In order to model message handling and cryptography, we define an inference system which formalizes the way messages are manipulated. An *inference system IS* is a set of

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

[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 [20]).

inference schemata (or rules) of the form

$$\frac{m_1 \quad \dots \quad m_n}{m}$$

where $m_1, \dots, m_n$ is a (possibly empty) set of *premises* and m is the *conclusion*. We say that m is *deducible* from a set of messages ϕ (and write $\phi \vdash m$) if we can obtain it by starting from ϕ and applying the inference rules. Each inference system induces a *deduction function* $\mathcal{D}(\phi) = \{m \mid \phi \vdash m\}$.

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

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

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

In Figure 1, we present a formalization (as inference system) of a simple deduction system which 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 do not allow any kind of cryptographic attack, e.g., the guessing of secret keys. However, since our semantics is parametric with respect to the deduction system used, we could also refine it in order to obtain more complex deductions which also consider some weaknesses of cryptographic algorithms.

The operational semantics of CSPA can be given in the usual way through *labelled transition systems* (LTS), which are automata with possibly infinitely many states. In particular some rules are given that make it possible to generate an LTS corresponding to every CSPA process and representing its operational behaviour. The details of the semantics are given in Appendix A.

Example 1. We present an example of a (trivial) protocol where A sends a message m_A to B encrypted with a key k_{AB} shared between A and B, then B shows the received message over channel *out*. [3]

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

where $k_{AB}^{-1} = k_{AB}$ (symmetric encryption) and $Msg(c) = \{\{m\}_k \mid m \in M, k \in K\}$ declaring the "type" of messages sent over c. We want to analyze the execution of P with no intrusions,

[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 $[\langle m_1 \dots m_r\rangle \vdash_{rule} x]E; \underline{0}$.

we thus consider $P \backslash \{c\}$, 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 $A(m_A, k_{AB}) \stackrel{\overline{c}\,\{m_A\}_{k_{AB}}}{\longrightarrow} \underline{0}$ and $B(k_{AB})$ can synchronize on that action by executing a $B(k_{AB}) \stackrel{c(\{m_A\}_{k_{AB}})}{\longrightarrow} [\langle \{m_A\}_{k_{AB}}, k_{AB}\rangle \vdash_{dec} z]\overline{out}\, z$. So

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

In the next section we will show that, in order to observe the behaviour of this protocol when executed in a hostile environment, we must use the knowledge restriction operator $(E)_\phi$ which forbids the guessing of secret values like m_A and k_{AB}, thus forcing a correct behaviour of the hostile environment with respect to cryptography. ■

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 Deductibility on Compositions* (NDC, for short) [9]. The main idea is the following: a system E is $GNDC^{\alpha}_{\lhd}$ *iff* for every environment X the composition of the system with X satisfies a specification $\alpha(E)$. Essentially, $GNDC^{\alpha}_{\lhd}$ guarantees that the property identified by α is satisfied (with respect to $\lhd$ relation) even when the system is composed with whatever 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 $GNDC^{\alpha}_{\lhd}$ 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 CSPA *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 knows every secret, it can basically simulate every message exchanged in the protocol. Indeed, when we model cryptographic protocols, it is necessary 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 through the knowledge restriction operator. 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). We call $\mathcal{E}_C^{\phi_X}$ the set of processes which can communicate on a subset of C and have an initial knowledge bound by ϕ_X:

$$\mathcal{E}_C^{\phi_X} = \{(E)_\phi \mid E \in \mathcal{E}_C \text{ and } \phi \subseteq \phi_X\}$$

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 in 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 execute $\overline{c}\,\{m_X\}_{k_{AB}}$ before getting any input, and this is not allowed to any process of the form $(E)_{\phi_X}$ if k_{AB} is not in ϕ_X.

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 1. *E is $GNDC^{\alpha}_{\lhd}$* iff

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

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

We propose a sufficient criterion for a static characterization (i.e. not involving the universal predicate $\forall$) of $GNDC^{\alpha}_{\lhd}$ 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:*

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

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

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 with 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*

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

3.2 Trace Equivalence

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

Definition 2. *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 $\gamma = \alpha_1 \ldots \alpha_n \in (Act \setminus \{\tau\})^*$ be a sequence of actions; then $E \stackrel{\gamma}{\Longrightarrow} E'$ if and only if there exist $E_1, E_2, \ldots,$ $E_{n-1} \in \mathcal{E}$ 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 3. *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 $T(E) = T(F)$.* $\blacksquare$

Trace preorder $\leq_{trace}$ is a pre-congruence with respect to value-passing CCS operators. It is easy to see that it is also a pre-congruence with respect to knowledge restriction.

Lemma 1. *Let $\phi \subseteq \mathcal{M}$. If $E \leq_{trace} F$ then $(E)_\phi \leq_{trace} (F)_\phi$.*

Through the knowledge restriction operator we can obtain in a natural way the most powerful intruder in the trace setting. We need the following definition:

$$Top_{trace}^L \stackrel{\text{def}}{=} \sum_{\substack{a \in Act \\ chan(a) \in L}} a.Top_{trace}^L$$

where L is a set of channels. Consider now process Top_{trace}^C which can execute every possible input and output on the set of public channels C. It is possible to prove that Top_{trace}^C is indeed the top of the set $\mathcal{E}_C^{\phi_X}$ with respect to $\leq_{trace}$:

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

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 $\alpha : \mathcal{E} \to \mathcal{E}$ we have that:*

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

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 he 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 also in the presence of whatever 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:

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

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 possible to analogously proof 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 $k_{AB} \notin \phi_X$. $\blacksquare$

4 An Example: The Wide Mouthed Frog Protocol

In this section we show how to use the GNDC properties to analyze a a simplified version of the Wide Mouthed Frog Protocol [3].

We consider two processes A and B respectively sharing keys k_{AS} and k_{BS} with a trusted server S. In order to establish a secure channel with B, A sends a fresh key k_{AB} encrypted with k_{AS} to the server S. Then, the server decrypts the key and forwards it to B, this time encrypted with k_{BS}. Now B has the key k_{AB} and A can send a message m_A encrypted with k_{AB} to B. The protocol should guarantee that when B receives m_A, such a message has been indeed originated by A. The protocol is composed of the following three messages:

$$\begin{array}{ll} \text{Message 1 } A \to S: & A, k_{AS}(B, k_{AB}) \\ \text{Message 2 } S \to B: & k_{BS}(A, k_{AB}) \\ \text{Message 3 } A \to B: & k_{AB}(m_A) \end{array}$$

The main difference with respect to the original protocol is that here messages 1 and 2 do not contain timestamps. This makes the protocol sensitive to a replay attack (as already remarked in [1]). We specify the protocol in CSPA as follows: [5]

$$\begin{aligned}
A(m,k) &\stackrel{\text{def}}{=} [\langle (B,k), k_{AS}\rangle \vdash_{enc} x]\ \overline{c_1}\,(A,x).\\
&\quad\ [\langle m,k\rangle \vdash_{enc} x]\ \overline{c_3}\,x\\
B &\stackrel{\text{def}}{=} c_2(y).\\
&\quad\ [\langle y, k_{BS}\rangle \vdash_{dec} z]\ [z \vdash_{snd} s]\ c_3(t).\\
&\quad\ [\langle t, s\rangle \vdash_{dec} w]\ \overline{out}\,w\\
S &\stackrel{\text{def}}{=} c_1(x).[\langle x, k_{AS}\rangle \vdash_{dec} y]\\
&\quad\ [y \vdash_{snd} z]\ [\langle (A,z), k_{BS}\rangle \vdash_{enc} k]\ \overline{c_2}\,k.\\
&\quad\ S\\
P &\stackrel{\text{def}}{=} A(m_A, k_{AB}) \,\|\, A(m'_A, k'_{AB}) \,\|\, B \,\|\, B \,\|\, S
\end{aligned}$$

where c_1, c_2 and c_3 are the three channels where messages 1,2 and 3 are communicated, respectively. Note that we have considered two instances of A and B in order to observe the reply attack. Indeed, if we consider only one instance of A and B no attack is possible here. Note also that A has has different messages and session keys in the two sessions.

Given this specification we would like to be guaranteed that even in the presence of the enemy we always observe the same messages on channel *out*. As a matter of fact, if this is true we can conclude that the enemy is not able to fake messages m_A and m'_A. [6] It is easy to see that P with no enemy, i.e., $P \setminus \{c_1, c_2, c_3\}$, is trace equivalent to process $\overline{out}\,m_A \,\|\, \overline{out}\,m'_A$. As a matter of fact the two session can be executed in any possible interleaving. This represent the "intended" execution of the protocol. Hence, we can check whether $P \in GNDC^{\overline{out}\,m_A \,\|\, \overline{out}\,m'_A}_{\leq_{trace}}$ or not. Unfortunately, this is not the case since we can show an enemy X such that: $P \,\|_C (X)_{\phi_X} \not\leq_{trace} \overline{out}\,m_A \,\|\, \overline{out}\,m'_A$, with $\phi_X = \emptyset$. It is the following process:

$$X \stackrel{\text{def}}{=} c_2(x).\overline{c_2}\,x.\overline{c_2}\,x.c_3(y).\overline{c_3}\,x.\overline{c_3}\,x$$

[5] For the sake of readability we directly write (y,z) in E instead of $[\langle y,z\rangle \vdash_{pair} x]E$. For example, $[\langle y,z\rangle \vdash_{pair} x]\overline{c}\,x$ becomes $\overline{c}\,(y,z)$. Moreover, this protocol specification is quite simplified since there are only two users and the possible sessions are fixed in advance. However, this is sufficient here to show how the attack can be revealed.

[6] This property is the NDC one [9, 10] which requires that the enemy cannot interfere at all with the protocol execution. This seems to be equivalent to the authentication property defined in [2].

which basically intercepts messages 2 and 3 and replays them. This enemy is indeed able to carry out the following attack:

Message 1	$A \to S$:	$A, k_{AS}(B, k_{AB})$	
Message 2	$S \to B$:	$k_{BS}(A, k_{AB})$	E eavesdrops and replays this
Message 2′	$E(S) \to B$:	$k_{BS}(A, k_{AB})$	
Message 3	$A \to B$:	$k_{AB}(m_A)$	E eavesdrops and replays this
Message 3′	$E(A) \to B$:	$k_{AB}(m_A)$	

Note that B concludes the protocol by receiving m_A twice. Formally, what we obtain is that $P \,\|_C\, X$ is able to execute the trace $\overline{out}\, m_A.\overline{out}\, m_A$ that is not a trace for process $\overline{out}\, m_A \,\|\, \overline{out}\, m'_A$. Hence, $P \,\|_C\, X \not\leq_{trace} \overline{out}\, m_A \,\|\, \overline{out}\, m'_A$ and so $P \notin GNDC^{\overline{out}\, m_A \,\|\, \overline{out}\, m'_A}_{\leq_{trace}}$.

5 Conclusions

In this paper we have proposed a uniform method for defining computer security properties which are based on a completely secure model of cryptography. In doing so, we have tried to exploit the common underlying ideas of existing proposals rather than proposing a completely new approach. Our aim was to find a quite flexible and useful schema that could help in reasoning about different properties. In [11] we show that our proposal is quite flexible 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 us to study the relationships among different proposals. For example, in [11] we have show that Non-Interference may be seen as the stronger property definable for a certain protocol since it requires that the action of whatever 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, e.g. anonymity [23].

Moreover, 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 definition by simply instantiating them. For example, we have shown that the static characterization result that permits to check a property only against the most general intruder, can be easily applied to all the trace-semantics based properties. This formally proves that the "most general intruder" approach followed in many verification tools (see, e.g., [14, 6]) is indeed correct. Hence such tools can also been used to verify GNDC-like properties over particular classes of processes (typically, processes with a finite LTS).

Another interesting point is that the *GNDC* schema itself suggests new analysis techniques. In [16, 17] an automated methodology for the analysis of some GNDC like properties is developed by starting from concepts used in concurrency theory. This approach does not require the specification of a fixed general intruder.

References

1. M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proceedings of 8th International Conference on Concurrency Theory (CONCUR '97)*, pages 59–73. Springer-Verlag, LNCS 1943, 1997.
2. M. Abadi and A. D. Gordon. "A Calculus for Cryptographic Protocols: The Spi Calculus". *Information and Computation 148*, 1:1–70, January 1999.
3. M. Burrows, M. Abadi, and R. Needham. "A Logic of Authentication". *Proceedings of the Royal Society of London*, 426:233–271, 1989.
4. J. Clark and J. Jacob. "A Survey of Authentication Protocol Literature: Version 1.0". Available at http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz, November 1997.
5. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.
6. A. Durante, R. Focardi, and R. Gorrieri. "CVS: A Compiler for the Analysis of Cryptographic Protocols". To appear in *Proceedings of Twelve IEEE Computer Security Foundation Workshop* (CSFW'99), June 1999.
7. F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman. Strand Spaces: Why is a Security Protocol Correct? In *Proceedings of the 1998 Symposium on Security and Privacy.* IEEE Computer Society Press, May 1998.
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 (H. Orman and C. Meadows Ed.)*, DIMACS Center, Rutgers University, September 1997. http://dimacs.rutgers.edu/Workshops/Security/.
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, September 1997.
11. R. Focardi and F. Martinelli. "A Uniform Approach for the Definition of Security Properties". Submitted for publication, 1999.
12. J. A. Goguen and J. Meseguer. Security Policy and Security Models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, April 1982.
13. R. Kemmerer, C. Meadows, and J. Millen. Three system 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 Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, pages 146–166, Passau (Germany), March 1996. Springer-Verlag, LNCS 1055.
15. G. Lowe. A Hierarchy of Authentication Specification. In *Proceedings of the 10th Computer Security Foundation Workshop*, pages 31–43. IEEE press, 1997.
16. F. Martinelli. Languages for Description and Analysis of Authentication Protocols. In *Proceedings of Sixth Italian Conference on Theoretical Computer Science (ICTCS'98)*, 1998.
17. F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties. In *Proceedings of the 11th Computer Security Foundation Workshop, (CSFW'98).* IEEE press, 1998.
18. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
19. J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-State Analysis of SSL 3.0. In *7th USENIX Security Symposium*, 1998.
20. S. Schneider. Security Properties and CSP. In *Proceedings of the 1996 Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, May 1996.
21. S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of The 11th Computer Security Foundations Workshop.* IEEE Computer Society Press, 1998.
22. S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), September 1998.
23. S. Schneider and A. Sidiropoulos. CSP and Anonymity. In *Proceedings of the Fourth ESORICS*, LNCS, pages 198–218, 1996.

A The Semantics of CSPA

The semantics of CSPA 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 4. *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),*
- *$\rightarrow \subseteq S \times T \times S$ 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.

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

$$(internal)\frac{}{\tau.E \xrightarrow{\tau} E} \qquad (+_1)\frac{E \xrightarrow{a} E'}{E + E_1 \xrightarrow{a} E'}$$

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

$$(\backslash L)\frac{E \xrightarrow{a} E' \quad chan(a) \notin L}{E \backslash L \xrightarrow{a} E' \backslash L} \qquad ([f])\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$$

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

$$(def)\frac{E[v_1/x_1, \ldots, v_n/x_n] \xrightarrow{a} E' \quad A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E}{A(v_1, \ldots, v_n) \xrightarrow{a} E'}$$

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

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

$$(\phi_1)\frac{E \xrightarrow{c(m)} E'}{(E)_\phi \xrightarrow{c(m)} (E')_{\phi \cup \{m\}}} \qquad (\phi_2)\frac{E \xrightarrow{\overline{c}\,m} E' \quad m \in \mathcal{D}(\phi)}{(E)_\phi \xrightarrow{\overline{c}\,m} (E')_\phi}$$

Fig. 2. Operational semantics.

The formal behaviour of a CSPA 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 CSPA 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 CSPA LTS reachable from the initial state E. Note that the rules that give the semantics of the two new operators $(E)_\phi$ and $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2$ are ϕ_1, ϕ_2 and $\mathcal{D}_1$, $\mathcal{D}_2$, respectively.