In *Proceedings of AMAST 2000*, Springer LNCS 1816, May 2000.

Message Authentication through *Non Interference* ⋆

Riccardo Focardi[1], Roberto Gorrieri[2], and Fabio Martinelli[3]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy.
e_mail: `focardi@dsi.unive.it`
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.
e_mail: `gorrieri@cs.unibo.it`
[3] Istituto per le Applicazioni Telematiche C.N.R., Pisa, Italy.
e_mail: `Fabio.Martinelli@iat.cnr.it`

Abstract. Authentication is a slippery security property that has been formally defined only recently; among the recent definitions, a rather interesting one has been proposed for the spi-calculus in [1, 2]. On the other hand, in a recent paper [10], we have proved that many existing security properties can be seen uniformly as specific instances of a general scheme based on the idea of non interference. The purpose of this paper is to show that, under reasonable assumptions, also spi-authentication can be recast in this general framework, by showing that it is equivalent to the non interference property called NDC of [8, 9].

1 Introduction

Authentication is, intuitively, the process of reliably verifying the identity of someone or something. This is usually achieved by some form of binding; e.g., a person is recognized on the net by some information (e.g., password or key) (s)he only knows. Authentication has, however, many facets. For instance, while *entity* authentication refers to the capability of identifying the other partner engaged in a session, *message* authentication usually refers to authenticity of both the origin and the content of the message.

Even if there is a widespread agreement on what authentication should be, under a closer scrutiny one realizes that it is a very slippery security property. As a matter of fact, formal definitions of authentication have rarely been given, not widely agreed upon, usually not compared and only recently proposed in the literature (see, e.g., [4, 12, 14, 22]). This is sometimes due to the fact that we first need a formal model on which the problem is defined (and this is often a source of possible proliferation of different proposals) and then a formal definition w.r.t. the chosen model. Moreover, even when a formal definition is given, usually this

⋆ Work partially supported by MURST Progetto "Certificazione automatica di programmi mediante interpretazione astratta" and Progetto "Teoria della Concorrenza, Linguaggi di Ordine Superiore e Strutture di Tipi"; CNR Progetto "Modelli e Metodi per la Matematica e l'Ingegneria"; CSP Progetto "ISA: Isp Secured trAnsactions".

is not (easily) comparable to others, due to different mathematical assumptions of the model.

The main aim of our current research is to find a uniform approach for defining the many variants of security properties (authentication in particular) in such a way that they can be all seen as specific instances of a general scheme (called GNDC [10]). This is badly needed in order to compare, classify and evaluate the merits of the various definitions and, possibly, to provide general, effective proof techniques that can be applied suitably for all properties.

To this aim, in [10] we have presented a process algebra, called CryptoSPA (in turn an improvement of SPA [9] which borrows some concepts from the language defined in [17]), that is expressive enough to model a large class of systems, e.g., (non mobile) security protocols. CryptoSPA has been chosen as the common model for comparing the various properties through the general, unifying scheme. The idea behind is essentially *non interference*, proposed many years ago [11] in a completely different context to study information flow in computer systems and widely studied in [8, 9, 18]. Roughly, a system is secure if its behaviour cannot be significantly altered (hence, with no interference) when executed in a hostile environment. This property is a direct generalization of *Non deducibility on composition* (NDC for short) we have proposed in [8, 9].

Some security properties (e.g., authentication as in [14] and denial of service as in [21]) have been shown as instances of our general scheme in [10]. The main goal of this paper is to show that also the rather different authentication property defined by Abadi and Gordon in [2], once adapted for CryptoSPA, can be formulated in our framework under some reasonable, mild assumptions. This is interesting for the following reasons:

- It strengthens our claim that non interference plays an important role in the specification and analysis of security protocols. Indeed, non interference seems the strongest property that can be defined for cryptographic protocols (see also [10]).
- It helps comparing the various authentication properties among them and with respect to other different security properties, as they are now defined uniformly (in the same language) as instances of the same general scheme.
- It contributes to clarify the spi authentication property: in fact, in its original definition, based on a testing-like semantics, the tester plays both the role of intruder and observer at the same time; in its new equivalent formulation, the two roles are clearly separated. Moreover, the new formulation does not require the explicit definition of the secure specification.
- It may contribute with proof techniques for spi-authentication, as there are already available techniques, also implemented in existing tools [6, 9], for NDC, that is at the base of the corresponding formulation.

2 The Model

In this section we report from [10] the language we use for the specification of authentication properties and protocols. It is called *Cryptographic Security*

Process Algebra (CryptoSPA for short), and it is basically a variant of value-passing CCS [19], where the processes are provided with some primitives for manipulating messages. In particular, processes can perform message encryption and decryption, and also construct complex messages by composing together simpler ones.

2.1 The CryptoSPA Syntax

CryptoSPA syntax is based on the following elements:

- A set $I = \{c_1, c_2, \ldots\}$ of *input* channels, a set $O = \{\bar{c}_1, \bar{c}_2, \ldots\}$ 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 $\mathcal{M}$;
- A set $C \subseteq I \cup O$ (s.t. $c \in C$ iff $\bar{c} \in C$) of *public* channels; these channels represent the insecure network where the enemy can intercept and fake messages;
- 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 along such a channel. Msg is such that $Msg(c) = Msg(\bar{c})$.
- A set $Act = \{c(m) \mid c \in I, m \in Msg(c)\} \cup \{\bar{c}m \mid \bar{c} \in O, m \in Msg(c)\} \cup \{\tau\}$ 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 $\bar{c}m$, 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:

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

where x is a variable, $m_1, \ldots, m_n$ are messages, $e, e_1, \ldots, e_r$ are messages (possibly containing variables) 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 the 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.

Besides the standard value-passing CCS operators, we have an additional one 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 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.

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

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

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

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

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

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

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

$$(\mathcal{D}_1)\frac{\langle m_1 \ldots m_r \rangle \vdash_{rule} m \quad E_1[m/x] \xrightarrow{a} E_1'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_1'} \qquad (\mathcal{D}_2)\frac{\nexists m : \langle m_1 \ldots m_r \rangle \vdash_{rule} m \quad E_2 \xrightarrow{a} E_2'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_2'}$$

Fig. 2. Operational semantics (symmetric rules for $+_1$, $\|_1$ and $\|_2$ are omitted).

2.2 The Operational Semantics of CryptoSPA

In order to model message handling and cryptography, in Figure 1 we define an inference system which formalizes the way messages may be manipulated by processes. It is indeed quite similar to those used by many authors (see, e.g., [13, 16]). 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 $\vdash_{dec}$). We denote with $\mathcal{D}(\phi)$ the set of messages that can be deduced by applying the inference rules on the messages in ϕ. Note that we are assuming encryption as completely reliable. Indeed we do not allow any kind of cryptographic attack, e.g., the guessing of secret keys. This 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 *labelled transition system* $< \mathcal{E}, Act, \{\xrightarrow{a}\}_{a \in A} >$, where $\xrightarrow{a}_{a \in A}$ is the least relation between CryptoSPA terms induced by axioms and inference rules of Figure 2.

2.3 Hostile Environments

In this section we report the characterization of hostile environments as given in [10]. Such a characterization is necessary to analyze protocols where some information is assumed to be secret, as it always happens in cryptographic protocols.

Basically, a *hostile environment* is an agent which tries to attack a protocol by stealing and faking the information which is transmitted on the CryptoSPA *public* channels in set C. In principle, such an agent could be modeled as a generic process X which can communicate only through the channels belonging to C. However, in this way we obtain that X is a completely powerful attacker which is able to "guess" every secret information and is thus not suitable when analyzing cryptographic protocols. This problem can be solved 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 $\phi_I \subseteq \mathcal{M}$ be the initial knowledge that we would like to give to the intruders X, i.e., the public information such as the names of the entities and the public keys, plus some possible private data of the intruders (e.g., their private key or nonces). For a certain intruder X, we want that all the messages in $ID(X)$ are deducible from ϕ_I. We thus define the set $\mathcal{E}_C^{\phi_I}$ of hostile processes as $\mathcal{E}_C^{\phi_I} = \{X \mid sort(X) \subseteq C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_I)\}$.

2.4 Semantic Equivalences

In this subsection we define some semantic equivalences that we will use to formalize some security properties. These equivalences equate processes which have an *indistinguishable behaviour*, depending on what we are interested to observe of the processes.

Trace Equivalence: Most of the security properties that have been proposed for the analysis of security protocols are based on the simple notion of *trace*: two processes are equivalent if they exactly show the same execution sequences (called *traces*). We need a transition relation $E \stackrel{\alpha}{\Longrightarrow} E'$ which does not consider internal τ moves. It is a shorthand for $E(\stackrel{\tau}{\longrightarrow})^* E_1 \stackrel{\alpha}{\longrightarrow} E_2(\stackrel{\tau}{\longrightarrow})^* E'$. For a trace $\gamma = \alpha_1 \ldots \alpha_n$ we write $E \stackrel{\gamma}{\Longrightarrow} E'$ if $E \stackrel{\alpha_1}{\Longrightarrow} E_1 \stackrel{\alpha_2}{\Longrightarrow} \cdots \stackrel{\alpha_{n-1}}{\Longrightarrow} E_{n-1} \stackrel{\alpha_n}{\Longrightarrow} E'$ for some $E_1, \ldots, E_{n-1}$. The set $T(E)$ of *traces associated with* E is then defined as $T(E) = \{\gamma \in (Act \setminus \{\tau\})^* \mid \exists E' : E \stackrel{\gamma}{\Longrightarrow} E'\}$.

Definition 1. *We write $E \leq_{trace} F$ if $T(E) \subseteq T(F)$. E and F are* trace equivalent *(notation $E \approx_{trace} F$) iff $E \leq_{trace} F$ and $F \leq_{trace} E$.* ■

Testing equivalence for CryptoSPA: In this section we give a notion of behavioural equivalence which incorporates the idea of *unguessable secrets* discussed in the previous section. This equivalence will allow us to rephrase in our model the notion of authentication used in the spi-calculus [1, 2]. The underlying idea is that two processes are equivalent if and only if they cannot be distinguished by any process that does not know the secret values. It is basically a weaker version of classical testing equivalence where equivalent processes should be indistinguishable by *any* environment [5].

We define the notion of experiment as given in the spi-calculus. A *test* is a couple (T, β), where T is a process called tester and β is a *barb*, i.e., a channel c or $\bar{c}$. We say that a process P *immediately passes a test* (T, β) (denoted by $P \,\|\, T \downarrow_\beta$) iff for some message m we have $P \,\|\, T \xrightarrow{\beta m}$. We also say that a process P *passes a test* (T, β) (denoted by $P \,\|\, T \Downarrow_\beta$) iff we have that $P \,\|\, T (\xrightarrow{\tau})^* P' \,\|\, T'$ and $P' \,\|\, T' \downarrow_\beta$.

We parameterize the notion of equivalence by the set of messages ϕ_I which are supposed to be known by the testers. Thus, we obtain the following set of possible testers: $\mathcal{E}^{\phi_I} = \{X \mid X \in \mathcal{E} \text{ and } ID(X) \subseteq \mathcal{D}(\phi_I)\}$. Note that it is strictly larger that the set of hostile environments $\mathcal{E}_C^{\phi_I}$ defined in the previous section. Indeed, such hostile environments can only communicate over the public channels in C.

Definition 2. $P \leq_{may} Q$ *iff* $\forall T \in \mathcal{E}^{\phi_I} : P \,\|\, T \Downarrow_\beta$ implies $Q \,\|\, T \Downarrow_\beta$. ■

We say that a process P is *may-testing equivalent* to a process Q iff $P \leq_{may} Q$ and $Q \leq_{may} P$. We denote that by $P \simeq Q$. It is easy to prove that $\simeq$ is an equivalence relation.

Testing equivalence has been exploited in a very elegant way for the formal analysis of security protocols in the spi-calculus as it implicitly provides a check over every possible hostile environment. In fact, a test can play at the same time the role of the attacker and the role of the observer that checks the outcomes of this attack. Thus, if two processes are may-testing equivalent, this means that they behave in the same way also when they are executed in a hostile environment.

Indeed, our notion of may-testing equivalence differs from the definition given for the spi-calculus, where a test can be every possible spi-calculus process. This difference is due to how secret messages are dealt with. Indeed, the restriction operator of the (s)pi-calculus supports the specification of *new*, an thus unguessable, messages directly inside a process. In this way, a special quantification over a subset of testers (with knowledge limited by ϕ_I) is not needed, and a classic testing-equivalence suffices. [1] The following holds:

Proposition 1. $P \leq_{trace} Q$ *implies* $P \leq_{may} Q$. ■

This derives from the standard result that classical may-testing preorder (with the quantification over any possible test) corresponds to trace preorder. It is interesting to observe that the opposite implication does not hold. As a counterexample consider a process $A(m, k) = [\langle m, k\rangle \vdash_{enc} x]\bar{c}x$ that encrypts m with k and sends it out on channel c. If k_A is secret (and also all the messages encrypted with k_A) then $A(M_1, k_A) \simeq A(M_2, k_A)$ even if $M_1 \neq M_2$.

On the other hand, if $M_1 \neq M_2$ then we have $A(M_1, k_A) \not\approx_{trace} A(M_2, k_A)$ because $\bar{c}\{M_1\}_{k_A}$ is a trace for $A(M_1, k_A)$ but not for $A(M_2, k_A)$.

[1] It is worthwhile noticing that in [15, 17] a language similar to CryptoSPA is indeed equipped with a form of secret generation.

3 Message Authentication in the spi-calculus

In [1, 2] an interesting notion of authentication is proposed. Consider a protocol $S(M)$, which has the aim of transmitting message M from one party A to another one B. $S(M)$ is guaranteeing authentication of message M if whenever a message m is delivered to B by $S(M)$, then m must be the same message as M.

The basic idea behind the verification of this property is the following: we have to generate from $S(M)$ a specification $S_{spec}(M)$ which guarantees authentication by construction, i.e., where the only message that can be delivered to B is indeed M. Then, it is verified whether $S(M)$ is may-testing equivalent to the specification $S_{spec}(M)$, i.e., whether there exists or not an attacker which can induce $S(M)$ to behave wrongly with respect to the delivery of M.

In [1, 2] a general method for generating the specification $S_{spec}(M)$ is not provided and the construction of $S_{spec}(M)$ is indeed illustrated by several example protocols. Intuitively, it can be obtained as follows: in the protocol $S(M)$, every time B is accepting a message m, such a message is replaced by M. In the following we formalize this intuition obtaining a general method for the construction of $S_{spec}(M)$ starting from $S(M)$. We feel that our formalization is general enough to model the significant examples of cryptographic protocols reported in [1, 2] (see also Section 6).

We assume that $S(M)$ is always composed of two parts: one strictly concerning the protocol *execution* and another one representing the *continuation* of the protocol. The latter is usually denoted by $F(x)$, where x represents the received message. The execution part can only send messages on public channels $c \in C$, while the continuation, by definition, does not take part in the protocol, and so it must use channels which are not in C. When may-testing equivalence is checked, the tester ideally acts as an attacker on the public channels and as an observer on the continuations. We give a simple protocol specified in this style.

Example 1. We specify a trivial protocol where Alice sends to Bob a message M as plaintext over an insecure channel c:

$$A(x) \stackrel{\text{def}}{=} \overline{c}x \qquad B \stackrel{\text{def}}{=} c(y).F(y) \qquad S(x) \stackrel{\text{def}}{=} A(x) \,\|\, B$$

When Bob receives the message in y, he just behaves like $F(y)$ which will possibly use the message in the future. The corresponding secure specification is:

$$A(x) \stackrel{\text{def}}{=} \overline{c}x \qquad B_{spec}(z) \stackrel{\text{def}}{=} c(y).F(z) \qquad S_{spec}(x) \stackrel{\text{def}}{=} A(x) \,\|\, B_{spec}(x)$$

where we can give to Bob a message z that is "magically" substituted for y in the continuation F. Indeed, in $S_{spec}(M)$ we have that Bob always behaves like $F(M)$ whatever he receives on channel c. This represents the secure version of $S(M)$, i.e., a version where the received message is always authentic by construction. Clearly, $S(M)$ does not guarantee any authentication of M: any external user can introduce a fake message M' on c that will be accepted by Bob. As a consequence, $S(M)$ can move to $F(M')$ while $S_{spec}(M)$ cannot.

Note that we can rewrite this protocol in a style that separates more evidently the execution part from the continuation one:

$$A(x) \stackrel{\mathrm{def}}{=} \overline{c}x \qquad B' \stackrel{\mathrm{def}}{=} c(y).\bar{p}y \qquad S'(x) \stackrel{\mathrm{def}}{=} (A(x) \,\|\, B' \,\|\, p(z).F(z)) \setminus p$$

As a matter of fact, $A(x) \,\|\, B'$ represents the execution, while $p(z).F(z)$ is the continuation with a "guard" $p(z)$ which has the special purpose of enabling the execution of $F(z)$ at the right time. It is possible to show that this form $S'(x)$ for the protocol is trace and may-testing equivalent to the initial one $S(x)$. ■

As we have seen in the example above it can be useful to write a protocol in a particular style that we call *normal form*. In general, more than one continuation could be present. Given a system S, we denote all of its occurrences of continuations as $\{F_1(x_1), \ldots, F_n(x_n)\}$, where x_i represents the only free variable of F_i.

We say that a system $S(m_1, \ldots, m_n)$ is in normal form if it is written as:

$$(P' \,\|\, \Pi_{i \in 1 \ldots n} p_{F_i}(x_i).F_i(x_i)) \setminus \boldsymbol{p} \tag{1}$$

where P' is the process S in which every continuation $F_i(x_i)$ is replaced by $\overline{p_{F_i}}(x_i)$, and $\boldsymbol{p} = \{p_{F_1}, \ldots, p_{F_n}\}$ is a set of channels that are used neither in S nor in F_i and are not contained in C. Note that the channels in $\boldsymbol{p}$ are indexed with the continuations F_i. This is useful for managing multiple concurrent sessions between senders and receivers which could be modeled by considering n copies of the sender and n copies of the receiver in parallel. [2]

An important assumption (common to the spi-calculus) over the continuations is that they should not know the protocol secrets. Indeed, in such a case they could help the tester attacking the protocol by making public such secrets.

Assumption 1 *We assume that for all continuations $F(x)$ and $\forall m \in \mathcal{D}(\phi_I)$ we have $F(m) \in \mathcal{E}^{\phi_I}_{Sort(F(m))}$.* ■

Given this particular form for the protocols, it is quite natural to derive a secure specification. Indeed, it is sufficient to define $S_{spec}(m_1, \ldots, m_n)$ as follows:

$$(P' \,\|\, \Pi_{i \in 1 \ldots n} p_{F_i}(x_i).F_i(m_i)) \setminus \boldsymbol{p} \tag{2}$$

Note that every continuation has its own correct message m_i. [3]

Remark 1. In the following, we will always consider systems in the normal form (1) and relative specifications in the form (2).

[2] Indeed, we implicitly assume that equal continuations (up to renaming of free variables) correspond to the same protocol between two users but in different sessions. See Section 6 for an example of multiple session protocol.

[3] Please also note that in the case of multiple sessions, this simply requires that a "correct" multiset of messages is delivered from one agent to another one, in whatever possible order. See Section 6 for an example.

Moreover, as done in the spi-calculus, we will always assume that the messages M_i to be delivered by the protocol are not secret, otherwise it would not be possible to observe them with testing equivalence.

Assumption 2 *When we consider a message M to be delivered by a protocol we always assume that $M \in \mathcal{D}(\phi_I)$.* ■

Hence, we can formalize in our framework the notion of authentication proposed in the spi-calculus.

Definition 3. *A system S guarantees weak spi−authentication iff for all vector of messages $(M_1, \dots, M_n)$ we have $S(M_1, \dots, M_n) \simeq S_{spec}(M_1, \dots, M_n)$.* ■

In [1], it is proposed another notion of authentication where there is also a quantification over the continuations F_i. In this way it is indeed possible to study if the protocol guarantees authentication in every possible context, rather than just in a special situation characterized by a specific continuation. To clarify this point we give the following simple example:

Example 2. Consider again the simple protocol of Example 1. Now, consider the special case where $F(x) = \mathbf{0}$. We obtain that for every M, $S(M)$ is equivalent to $S_{spec}(M)$. Indeed $S(M)$ and $S_{spec}(M)$ become syntactically the same process if $F(x) = \mathbf{0}$. However, we have already observed that this trivial protocol cannot guarantee the authenticity of M. On the contrary, if we replace $F(x)$ with $\overline{out}\ x$ then we may detect the attack. The continuations play thus a central role. In particular, if the behaviour of a continuation does not depend on the message x then it cannot help detecting authentication attacks. ■

Let $\boldsymbol{F}$ be a vector of continuations $\langle F_1, \dots, F_n \rangle$. We will denote with $S^{\boldsymbol{F}}$ the system S where its continuations are replaced by $\boldsymbol{F}$. Moreover, the initial vector of a system S will be denoted by $\boldsymbol{F}^*$. When we quantify over all the possible vector of continuations, we require that equal continuations are always replaced by equal continuations. More formally, we say that two vectors $\langle F_1, \dots F_n \rangle, \langle F_1', \dots, F_n' \rangle$ are *compatible* if and only if $\forall i, j \in \{1, \dots n\} : F_i = F_j$ iff $F_i' = F_j'$. We then consider only the vectors of continuations that are compatible with the initial one $\boldsymbol{F}^*$. This condition is necessary if we want to analyze multiple sessions where many instances (all with the same continuation) of an agent are considered (see Section 6). Hence, we obtain the following more general definition:

Definition 4. *A system S guarantees (strong) spi−authentication iff for all the vectors of continuations $\boldsymbol{F}$* compatible *with $\boldsymbol{F}^*$, and for all vectors of messages $(M_1, \dots, M_n)$ we have $S^{\boldsymbol{F}}(M_1, \dots, M_n) \simeq S_{spec}^{\boldsymbol{F}}(M_1, \dots, M_n)$.* ■

In the next section we will see that there exists a *canonical continuation* $\boldsymbol{F'}$ such that the weak spi-authentication of $S^{\boldsymbol{F'}}$ implies spi-authentication of S. The solution of using may-testing equivalence for authentication verification is indeed very elegant, but mixing the tester and the intruder could generate some confusion. Indeed, it would be much more intuitive to separate the intrusion activity from the equivalence check. This is what is done in the property we are going to present in the next section. Moreover, here the definition of the specification is somewhat arbitrary and unclear. In the next section we will show that the definition of a secure specification is not necessary anymore when we adopt a notion of authentication based on Non-interference.

4 NDC-based Authentication

We recall the notion of Non-Interference (NI) [11]. NI was proposed in system security as a model for the detection of all possible interferences from a certain group of users to another one. It has been formalized in different ways (see, e.g., [8, 11, 20, 23]), and also applied in the verification of various properties of security protocols [6, 7], among which authentication. Indeed, the correctness of a protocol can be proved by guaranteeing that an enemy is not able to interfere at all with the execution of the protocol, i.e., that the protocol with and without the enemy behaves exactly in the same way. In this respect, we have already noticed a similarity in the two approaches.

In the following we will use the generalization of NI to the process algebraic setting proposed in [8] and called *Non Deducibility on Compositions (NDC)*. The idea is the following: the group U_H of users (or high level processes) that must not interfere with the other users (in group U_L of low level processes) is characterized by the set of actions that its components can execute. Let C be the set of channels which processes in U_H can use for communication. A system S is NDC if every possible process $X \in U_H$ composed with S is not able to modify the behaviour of S observed from the point of view of the users in U_L. Thus, U_H corresponds to $\mathcal{E}_C^{\phi_I}$ and NDC can be defined as follows, where we always consider processes S which are in form (1).

Definition 5. *A process S is NDC iff $\forall X \in \mathcal{E}_C^{\phi_I} (S \,||\, X) \setminus C \approx_{trace} S \setminus C$.* ■

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

Note that NDC, in its original definition, is based on trace equivalence. Here we give a definition of may-testing based NDC (TNDC) in order to compare it with the authentication notion defined in the previous section.

Definition 6. *A process S is TNDC iff $\forall X \in \mathcal{E}_C^{\phi_I} (S \,||\, X) \setminus C \simeq S \setminus C$.* ■

An intuitive notion of authentication can be given in a natural way through TNDC/NDC as follows:

Definition 7. *Let S be a protocol. Then, S guarantees (T)NDC-authentication iff for all messages $M_1, \ldots, M_n$ we have that $S(M_1, \ldots, M_n) \in (T)NDC$.* ■

Actually in the definition above we do not take care of the continuations. In particular, $S(M_1, \ldots, M_n) \in (T)NDC$ requires that $S(M_1, \ldots, M_n)$ composed with whatever enemy $X \in \mathcal{E}_C^{\phi_I}$ is equivalent to $S(M_1, \ldots, M_n) \setminus C$. Since C represents the set of channels over which the parties communicate, then $S(M_1, \ldots, M_n) \setminus C$ corresponds to a secure specification (execution with protected channels). In this approach we obtain, in a sense, the secure specification for free.

It is interesting to see that since we restrict the behaviour of the system on the channels in C we actually do not observe directly the communication, but only the behaviour on the continuations. Indeed, we obtain a specification that is somewhat simpler than the one needed in the spi-calculus authentication approach. As an example, consider again the protocol $S(x)$ of Example 1. We obtain that $S(M) \setminus C$ is trace and may-testing equivalent to the process $F(M)$, which is just the expected (correct) continuation for the protocol.

5 Comparison

In this section, we formally compare the notions of authentication presented in the previous sections, i.e., the one proposed for the spi-calculus with the TNDC/NDC-based one.

We have already pointed out some of the similarities between them: both of the properties are based on a notion of behavioral equivalence; moreover, they both check whether the "system under attack" behaves like a secure specification. It is however important to notice that this is done in a quite different way. In the spi-calculus the system is implicitly checked against all the possible interactions with the (hostile) environment through the use of the may-testing equivalence. On the contrary, the TNDC/NDC-based approach does it explicitly through the quantification over possible attackers.

The first interesting result shows that authentication in the spi-calculus is at least as discriminating as the TNDC-based one, when S_{spec} guarantees TNDC-authentication.

Proposition 2. *Let S be a protocol and S_{spec} be a secure specification (for S) that guarantees TNDC-authentication. If S guarantees spi-authentication then S also guarantees TNDC-authentication.* ∎

One of the hypotheses of the proposition above is that S_{spec} guarantees TNDC-authentication. As a matter of fact, we can prove the results of this section, under the following well-formedness conditions:

(i) For all vector of messages $\boldsymbol{M}$, $S(\boldsymbol{M}) \setminus C \simeq S_{spec}(\boldsymbol{M}) \setminus C$
(ii) For all vector of messages $\boldsymbol{M}$, $S_{spec}(\boldsymbol{M}) \setminus C \simeq \Pi_{k \in 1..n} F_k(M_k)$

They are both very natural. In particular (i) is an obvious requisite for S_{spec}: the system S and its specification S_{spec} behave the same when the public channels are protected through the restriction (when no attack is possible). Condition (ii) requires that all the continuations of the specifications when there is no attacker at all, are eventually enabled, i.e., the specification is well-formed since it does not contain unreachable continuations. If this is not the case some useless redundancy is present in S_{spec}. By exploiting condition (ii), we can prove:

Lemma 1. *S_{spec} guarantees TNDC-authentication.* ■

This lemma gives, in a sense, a formal evidence of the correctness of how we define specifications S_{spec}. Thus, in general the spi-authentication is at least as discriminating as the TNDC-based one is:

Corollary 1. *If a system S guarantees spi-authentication then S also guarantees TNDC-authentication.* ■

The other implication is less obvious. As a matter of fact, since in TNDC the action of the intruder is separated from the observation activity, we have to be sure that this "splitting" can be performed without loosing discriminating power. As our intruders are not able to communicate over the channels of the continuations, one may think, at a first glance, that we may miss some attack. We show that this is not the case.

First of all, we simplify the notion of spi-authentication by defining *canonical* continuations, i.e., special continuations that allow us to avoid the universal quantification over all the possible vectors $\boldsymbol{F}$. In particular, we consider the special vector of continuations $\boldsymbol{F'}$ where $F'_i(x_i) = \overline{out_{F_i^*}}(x_i)$ and such that the channels $out_{F_i^*}$ are not public, i.e., are not in C. Please note that this vector is compatible with $\boldsymbol{F}^*$. By exploiting condition (i), we can prove:

Proposition 3. *We have that S guarantees $spi-authentication$ iff $S^{\boldsymbol{F'}}$ guarantees weak $spi-authentication$.* ■

This interesting result basically states that there exists a *canonical continuation* $\boldsymbol{F'}$ that can be considered in order to prove spi-authentication for every possible continuation. Moreover, this result can also be exploited to compare such notion of authentication with the NDC-based one. The following holds:

Lemma 2. *If $S^{F'}$ guarantees TNDC-authentication then $S^{F'}$ also guarantees weak spi-authentication.* ■

¿From the previous results it can be easily proved the following.

Theorem 1. *A system S guarantees (strong) spi-authentication if and only if $S^{\boldsymbol{F'}}$ guarantees TNDC-authentication.* ■

Now we show that the two notions of message authentication based on non interference, i.e. TNDC and NDC, actually coincide when we study canonical forms. This is very useful to avoid the implicit quantification in TNDC due to testing equivalence. We give the following proposition which can be applied when P is the product of the canonical continuations $\boldsymbol{F'}$ in order to obtain the final result (Theorem 2).

Proposition 4. *Let P be a CryptoSPA process which can only execute a finite number of output actions and such that $ID(P) \subseteq \mathcal{D}(\phi_I)$. Then, for all Q, $P \simeq Q$ iff $P \approx_{trace} Q$.* ■

Finally, the following result completes the comparison:

Theorem 2. *A system S guarantees (strong) spi-authentication if and only if $S^{\boldsymbol{F}'}$ guarantees NDC-authentication.* ∎

6 An Example: The Wide Mouthed Frog Protocol

In this section we show how to use NDC to analyze a a simplified version (also studied in [2]) 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:

$$\begin{array}{l}
\text{Message 1 } A \to S : \; A, \{B, k_{AB}\}_{k_{AS}} \\
\text{Message 2 } S \to B : \; \{A, k_{AB}\}_{k_{BS}} \\
\text{Message 3 } A \to B : \; \{m_A\}_{k_{AB}}
\end{array}$$

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 as the following CryptoSPA normal form: [4]

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

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 different messages and session keys in the two sessions.

In order to check if P satisfies (strong) spi-authentication we can just verify if $P^{\boldsymbol{F}'}$ guarantees NDC-authentication. Note that $P^{\boldsymbol{F}'}$ is obtained from P by replacing $F(w)$ with $\overline{out_F}\, w$. Note also that in the two instances of B the continuation is exactly the same (this allows to model multiple sessions). We find out that $P^{\boldsymbol{F}'}$ does not guarantee such a property. Indeed, it is easy to

[4] 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.

see that P with no enemy, i.e., $P \setminus \{c_1, c_2, c_3\}$, is trace equivalent to process $\overline{out_F}\, m_A \,\|\, \overline{out_F}\, m'_A$. As a matter of fact the two sessions can be executed in any possible interleaving. This represent the intended execution of the protocol. Now, consider the enemy $X \stackrel{\mathrm{def}}{=} c_2(x).\overline{c_2}\, x.\overline{c_2}\, x.c_3(y).\overline{c_3}\, y.\overline{c_3}\, y$. It is easy to see that $(P \,\|\, X) \setminus C$ is able to execute the trace $\overline{out_F}\, m_A.\overline{out_F}\, m_A$ that is not a trace for process $\overline{out_F}\, m_A \,\|\, \overline{out_F}\, m'_A$. Hence, $(P \,\|\, X) \setminus C \not\leq_{trace} \overline{out_F}\, m_A \,\|\, \overline{out_F}\, m'_A$ and NDC-authentication is not satisfied.

The enemy X basically intercepts messages 2 and 3 and replays them, inducing B to commit twice on message m_A (as shown by trace $\overline{out_F}\, m_A.\overline{out_F}\, m_A$). This attack is quite critical in some situations. As an example, m_A could be a request of money transfer that would be executed twice. In order to avoid this attack, it could be possible to modify the protocol (as done in [2]) by adding nonce handshakes. We do not show this here for lack of space.

7 Future Work

In order to compare various formalizations of security properties, we have defined in [10] a general scheme that permits to capture a number of properties (e.g., authentication as in [14] and denial of service as in [21]) as particular instances of the scheme itself. The results presented in this paper have allowed us to extend the set of properties defined in the scheme. Our main issue is now to find comparison results in order to obtain a complete classifications in the style of [8, 14], which could help in evaluating the relative merits of all such properties.

Another aim of our current research is to provide general, effective proof techniques that can be suitably applied to a set of security properties. In this paper, we have seen that the spi-authentication corresponds to a notion of authentication based on NDC. This permits, in principle, the reuse of automatic checking techniques for NDC in order to check spi-authentication over CryptoSPA protocols; indeed if ϕ_I is finite then it is possible to find a most-general intruder Top such that NDC is reduced to just one check $(S^{\boldsymbol{F}'}(M_1, \ldots, M_n) \,\|\, Top) \setminus C \approx_{trace} S^{\boldsymbol{F}'}(M_1, \ldots, M_n) \setminus C$ (see [10] for more details) that could be verified using the tool in [6]. The problem of finding a finite set of vectors of messages which could avoid the universal quantification over all the possible messages $\boldsymbol{M}$, is still open.

We are currently studying the relations between spi and NDC authentication also in the spi-calculus model, where the possibility of communicating channel names between processes gives rise to new interesting issues. These complicate very much the comparison, but we feel that at least some of the results presented here also hold in such a model.

References

[1] M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proc. 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] M. Burrows, M. Abadi, and R. Needham. "A Logic of Authentication". *Proc. of the Royal Society of London*, 426:233–271, 1989.

[4] C. Bodei, P. Degano, R. Focardi, and C. Priami. Authentication via localized names. In *Proc. 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] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proc. of CSFW'99*, pages 203–212. IEEE press, 1999.

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

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

[9] 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.

[10] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proc. of World Congress on Formal Methods (FM'99)*, pages 794–813. Springer, LNCS 1708, 1999.

[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] D. Gollmann. What do we mean by entity authentication? In *Proc. of Symposium in Research in Security and Privacy*, pages 46–54. IEEE Press, 1996.

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

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

[15] D. Marchignoli and F. Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In *Proc. of TACAS'99*, volume 1579 of *LNCS*, pages 148–163, 1999.

[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 *Proc. of ICTCS'98*, pages 304–315. World Scientific, 1998.

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

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

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

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

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

[23] J. T. Wittbold and D. M. Johnson. "Information Flow in Nondeterministic Systems". In *Proc. of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 144–161. IEEE Computer Society Press, 1990.