ELSEVIER

Theoretical Computer Science 291 (2003) 285–327

Theoretical Computer Science

www.elsevier.com/locate/tcs

A comparison of three authentication properties ☆

Riccardo Focardi[a], Roberto Gorrieri[b], Fabio Martinelli[c,*]

[a] *Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy*
[b] *Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy*
[c] *Istituto per le Applicazioni Telematiche C.N.R., Pisa, Italy*

Received 8 January 2001; received in revised form 6 August 2001; accepted 2 January 2002

Abstract

Authentication is a slippery security property that has been formally defined only recently; among the recent definitions, two rather interesting ones have been proposed for the spi-calculus by (Abadi and Gordon (in: Proc. CONCUR'97, Lecture Notes in Computer Science, Vol. 1243, Springer, Berlin, 1997, pp. 59–73; Inform. and Comput. 148(1) (1999) 1–70) and for CSP by Lowe (in: Proc. 10th Computer Security Foundation Workshop, IEEE Press, 1997, pp. 31–43). On the other hand, in a recent paper (in: Proc. World Congr. on Formal Methods (FM'99), Lecture Notes in Computer Science, Vol. 1708, Springer, Berlin, 1999, pp. 794–813), 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, spi-authentication can be recast in this general framework as well, by showing that it is equivalent to the non-interference property called NDC of Focardi and Gorrieri (J. Comput. Security 3(1) (1994/1995) 5–33; IEEE Trans. Software Eng. 23(9) (199) 550–571). This allows for the comparison between such a property and the one based on CSP, which was already recast under the general scheme of Focardi and Martinelli (1999).

Keywords: Security; Cryptographic protocols; Authentication; Non-interference; Process algebra

☆ Extended and revised version of [12]. Work partially supported by MURST projects "Interpretazione astratta, type systems e analisi control-flow", "Formal Methods for Security" (MEFISTO) and Microsoft Research Europe. The third author is also supported by a CSP grant for the project "SeTAPS".

* Corresponding author.

E-mail addresses: focardi@dsi.unive.it (R. Focardi), gorrieri@cs.unibo.it (R. Gorrieri), fabio.martinelli@iat.cnr.it (F. Martinelli).

URLs: http://www.dsi.unive.it/∼focardi, http://www.cs.unibo.it/∼gorrieri

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00365-1

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., users are recognized on the net by some information (e.g., password or key) only they know. Authentication has, however, many facets. For instance, in a communication protocol, *entity* authentication commonly refers to the capability of identifying the other party engaged in a protocol session, while *message* authentication usually refers to authenticity of both the origin and the content of the exchanged 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., [3,16,18,27]). 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 of authentication w.r.t. the chosen model. Moreover, even when a formal definition is given, usually this 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 all be seen as specific instances of a general scheme. This is badly needed in order to compare, classify and evaluate the merits of the various definitions and possibly provide general and effective analysis techniques that can be applied suitably for all properties.

To this aim, in [14] we have presented a process algebra, called CryptoSPA,[1] 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, called GNDC. The main idea behind GNDC is the notion of *non-interference*, which was proposed many years ago [15] in a completely different context to study information flow in computer systems and was widely studied in [9,10,21]. Roughly, a system satisfies a security property if its behavior cannot be altered (hence, with no interference) when executed in a hostile environment. This property is a direct generalization for security protocols of the property of *non-deducibility on composition* (NDC for short) that we proposed in [9,10].

Some security properties (e.g., CSP authentication of [18] and non-repudiation as in [26]) have been shown as instances of our general scheme in [12,14]. The main goal of this paper is to show that the rather different authentication property defined by Abadi and Gordon [2], once adapted for CryptoSPA, can be formulated in our framework as well, under some reasonable, mild assumptions. This new formulation of the property defined in [2] is interesting for the following reasons:

[1] CryptoSPA is an improvement of (value-passing) SPA [10] which borrows some concepts for handling cryptography from the language defined in [22,20].

- It strengthens our claim that non-interference plays a key role in the specification and analysis of security protocols, as also the spi-authentication property can be recast in the same scheme.
- It helps in 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. For instance, here we show that in a particular (but reasonable) situation the *Agreement* authentication property defined in [18] is stronger than spi-authentication [2].
- 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 analysis techniques for spi-authentication: the new formulation is based on NDC and techniques for the verification of NDC are already available, also implemented in existing tools [6,7,10].

The paper is organized as follows: in Section 2 we define the model; in Section 3 we adapt to our model the notion of message authentication of the spi-calculus; Section 4 describes NDC-based authentication, while Section 5 describes the CSP authentication property called *Agreement*; Section 6 shows that, under some assumptions, message authentication of the spi-calculus and NDC-based authentication are equivalent; in Section 6 we also compare *Agreement* with spi-authentication, showing that the former implies the latter under some reasonable assumptions; Section 7 briefly discusses some verification issues; Section 8 describes some concluding remarks and future work. Finally, an appendix contains the proofs of some results reported in the paper.

2. The model

In this section we describe the language we use for the specification of authentication properties and protocols, originally presented in [14]. It is called *cryptographic security process algebra* (CryptoSPA for short), and is a variant of value-passing CCS [23], 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

The CryptoSPA syntax is based on the following elements:

- A set $I=\{a,b,\ldots\}$ of *input* channels and a set $O=\{\bar{a},\bar{b},\ldots\}$ of *output* channels, related through a function $\bar{\cdot}: I\cup O\to I\cup O$ which given an input $a\in I$ returns the *corresponding* output $\bar{a}\in O$ and vice versa, i.e., $\bar{\bar{a}}=a$.
- A set M of basic messages. The set $\mathcal{M}$ of all messages is defined as the least set such that $M\subseteq\mathcal{M}$ and $\forall m,m',k\in\mathcal{M}$ we have that (m,m') (pairs) and $\{m\}_k$ (encryptions) also belong to $\mathcal{M}$.

- A set $C \subseteq I \cup O$ of channels, ranged over by c, such that $c \in C$ iff $\bar{c} \in C$; these channels represent the insecure network on which the enemy can intercept and fake messages. Channels in $(I \cup O) \backslash C$ are the *private* channels.
- A function $Msg : I \cup O \to \mathscr{P}(\mathscr{M})$ 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 *Var* of *variables*, ranged over by x.
- A set *Const* of constants, ranged over by A.

The syntax of CryptoSPA terms (or processes) is defined as follows:

$$E ::= \underline{0} \mid c(x).E \mid \bar{c}e.E \mid \tau.E \mid E + E \mid E \,\|\, E \mid E\backslash L \mid E[f] \mid \\ \mid A(e_1, \ldots, e_n) \mid [e = e']E; E \mid [\langle e_1 \ldots e_r \rangle \vdash_{\text{rule}} x]E; E$$

where $e, e', e_1, \ldots, e_r$ are messages or variables, L is a set of input channels and $f : Act \mapsto Act$ is a function that relabels channel names inside actions.[2] Both the operators $c(x).E$ and $[\langle e_1 \ldots e_r \rangle \vdash_{\text{rule}} x]E; E'$ bind the variable x in E. It is also assumed that each constant A has an associated defining equation: $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E$ where E is a CryptoSPA process which may contain no free variables except $x_1, \ldots, x_n$, which must be distinct. Constants permit us to define recursive processes.

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 $[\langle m_1 \ldots m_r \rangle \vdash_{\text{rule}} x]E_1; E_2$ process tries to deduce a piece of information z from the tuple of messages $\langle m_1 \ldots m_r \rangle$ through one application of rule $\vdash_{\text{rule}}$; if it succeeds then it behaves like $E_1[z/x]$, otherwise it behaves like E_2. See next subsection for a more detailed explanation of derivation rules.

We call $\mathscr{E}$ the set of all the CryptoSPA closed terms (i.e., with no free variables), and we define $sort(E)$ to be the set of all the channels syntactically occurring in the term E. Moreover, for the sake of readability, we always omit the termination $\underline{0}$ at the end of process specifications, e.g., we write a in place of $a.\underline{0}$. We also write $[m = m']E$ in place of $[m = m']E; \underline{0}$ and analogously for $[\langle m_1 \ldots m_r \rangle \vdash_{\text{rule}} x]E; \underline{0}$. Finally, we often replace constructive rules (encryption and pairing) with the resulting messages, e.g., we use $\bar{c}\{m\}_k$ as a shortcut for $[\langle m, k \rangle \vdash_{\text{enc}} x]\bar{c}x$.

2.2. The operational semantics of CryptoSPA

In order to model message handling and cryptography, CryptoSPA may be equipped with a set of inference rules (inference system). Note that CryptoSPA syntax, its

[2] The relabeling functions map channels in C to channels in C and channels in $(I \cup O) \backslash C$ to channels in $(I \cup O) \backslash C$.

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

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

Fig. 1. Inference system for message manipulation, where $m, m', k, k^{-1} \in \mathscr{M}$.

semantics and the results obtained herein are completely parametric with respect to the inference system used. It is thus quite easy to adopt other rules, e.g., for modeling some kinds of cryptographic weakness. For explanatory purposes, in Fig. 1, we provide a simple inference system which is quite similar to those used by many authors (see, e.g., [17,19]). We consider a function $\cdot^{-1} : \mathscr{M} \to \mathscr{M}$ which denotes, for each key k (i.e., a message possibly used as encryption key), the corresponding decryption key. Note that there are no rules, to obtain the message k^{-1} from k (and vice versa). In particular, the inference system can combine two messages obtaining a pair (rule $\vdash_{\text{pair}}$); it can extract one message from a pair (rules $\vdash_{\text{fst}}$ and $\vdash_{\text{snd}}$); 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 $\vdash_{\text{enc}}$ and $\vdash_{\text{dec}}$). As an example, process $[\langle \{m\}_k, k^{-1}\rangle \vdash_{\text{dec}} x]E_1; E_2$ decrypts message $\{m\}_k$ through the inverse key k^{-1} and behaves like $E_1[m/x]$, while $[\langle \{m\}_k, k'\rangle \vdash_{\text{dec}} x]E_1; E_2$ (with $k' \neq k^{-1}$) tries to decrypt the same message with the wrong inverse key k' and (since it is not permitted by $\vdash_{\text{dec}}$) it behaves like E_2.

Given an inference system, we say that a message m can be deduced from a set of messages ϕ whenever there exists a tree whose nodes are messages, such that the root is m, the leaves are contained in ϕ and each message in the tree may be obtained by applying a rule instance of the inference system whose premises are the descendants of the message in the tree. We consider a function $\mathscr{D}$, from finite sets of messages to sets of messages, such that $\mathscr{D}(\phi)$ is the set of messages that can be deduced from ϕ. We assume that $\mathscr{D}$ is recursive.

Note that, in our model, we are assuming encryption as completely reliable. Thus, we do not allow any kind of cryptographic attack, e.g., the guessing of secret keys. Nevertheless, we observe the attacks that can be carried out even if cryptography is completely reliable.

The behavior of a CryptoSPA term is formally described by means of the *labeled transition system* $\langle \mathscr{E}, Act, \{\xrightarrow{a}\}_{a\in Act}\rangle$, where $\xrightarrow{a}_{a\in Act}$ is the least relation between CryptoSPA terms induced by the axioms and inference rules of Fig. 2.

Example 2.1. We present a very simple example of a protocol where A sends a message m_A to B encrypted with a key k_{AB} shared between A and B. We define it as $P \stackrel{\text{def}}{=} A(m_A, k_{AB}) \,\|\, B(k_{AB})$ where $A(x_1, x_2) \stackrel{\text{def}}{=} \bar{c}\{x_1\}_{x_2}$ and $B(x_1) \stackrel{\text{def}}{=} c(y).[\langle y, x_1\rangle \vdash_{\text{dec}} z]\overline{out}\, z$. Moreover, $k_{AB}^{-1} = k_{AB}$ (symmetric encryption) and $Msg(c) = \mathscr{M}$. We want to analyze the execution of P with no intrusion, we thus consider $P\backslash\{c\}$, since the restriction guarantees that c is now a private channel between A and B. We obtain a process whose

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

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

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

$$(\backslash L)\frac{E \xrightarrow{a} E' \quad chan(a) \notin L}{E\backslash L \xrightarrow{a} E'\backslash L} \qquad (\mathscr{D}_1)\frac{\langle m_1 \ldots m_r\rangle \vdash_{\text{rule}} m \qquad E_1[m/x] \xrightarrow{a} E_1'}{[\langle m_1 \ldots m_r\rangle \vdash_{\text{rule}} x]E_1; E_2 \xrightarrow{a} E_1'}$$

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

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

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

only possible execution is the correct one where A sends to B the message $\{m_A\}_{k_{AB}}$ and then $\overline{out}\,m_A$ is executed:

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

2.3. The enemy

In this section we characterize the crucial notion of *enemy* (or intruder) as done in [14]. Such a characterization is necessary to analyze protocols where some information is assumed to be secret, as it always happens in cryptographic protocols. Intuitively, an enemy can be thought of as a process 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 a process 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 too powerful attacker which is able to "guess" every secret information (e.g., the private key K_{AB} of Example 2.1), as illustrated in the following example.

Example 2.2. Consider again the protocol P of Example 2.1. Since only A and B know k_{AB}, this protocol should guarantee the authenticity of m_A even in the presence of an enemy. We assume that $c \in C$ is a public channel and we consider the following process:

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

This process may only communicate over the public channel c, as $sort(X(m,k))=\{c\}$. Consider now process $X(m_X, k_{AB})$, which knows k_{AB} and can consequently send a faked message $\{m_X\}_{k_{AB}}$ to B. In order to observe this, we consider the following process

$$
\begin{aligned}
I(\underline{0}, V) &= \emptyset \\
I(c(x).E, V) &= I(E, V) \\
I(\bar{c}\,e.E, V) &= \mathit{get\text{–}msg}(e) \cup I(E, V) \\
I(\tau.E, V) &= I(E, V) \\
I(E_1 + E_2, V) &= I(E_1, V) \cup I(E_2, V) \\
I(E_1 \parallel E_2, V) &= I(E_1, V) \cup I(E_2, V) \\
I(E \setminus L, V) &= I(E, V) \\
I(E[f], V) &= I(E, V) \\
I(A(e_1, \ldots, e_n), V) &= \begin{cases} \bigcup\limits_{i \in \{1,\ldots,n\}} \mathit{get\text{–}msg}(e_i) & \text{if } A \in V \\ I(E, V \cup \{A\}) \cup \bigcup\limits_{i \in \{1,\ldots,n\}} \mathit{get\text{–}msg}(e_i) & \\ & \text{otherwise} \end{cases} \\
& \quad \text{where } A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E \\
I([e = e']E_1; E_2, V) &= \mathit{get\text{–}msg}(e) \cup \mathit{get\text{–}msg}(e') \cup I(E_1, V) \cup I(E_2, V) \\
I([\langle e_1 \ldots e_r \rangle \vdash_{\text{rule}} x]E_1; E_2, V) &= \left(\bigcup_{i \in \{1,\ldots,r\}} \mathit{get\text{–}msg}(e_i) \right) \cup I(E_1, V) \cup I(E_2, V)
\end{aligned}
$$

where

$$
\mathit{get\text{–}msg}(e) = \begin{cases} \{e\} & \text{if } e \text{ is a message} \\ \emptyset & \text{if } e \text{ is a variable} \end{cases}
$$

Fig. 3. Definition of $I(E, V)$.

"under attack" (note that we put X inside the scope of restriction):

$$(P \parallel X(m_X, k_{AB}))\backslash\{c\}$$

After one τ communication step, the process above can perform $\overline{out}\, m_X$ which represents the fact that B has 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 such behavior since, as mentioned above, we are interested in attacks that can be carried out even when cryptography is completely reliable.

This problem of guessing secret values can be solved by imposing some constraints on the initial data known by the intruders. Given a process E, we call $ID(E)$ the set of messages that appear in E. More formally, we define $ID(E)$ as $I(E, \emptyset)$, where $I : \mathscr{E} \times \mathscr{P}(Const) \to \mathscr{P}(M)$ is given in Fig. 3. Informally, $I(E, V)$ is a function that recursively visits the sub-terms of E and the body of the constants used. The argument V is used to check that the unwinding of a constant definition is performed only once.

Example 2.3. Consider $A(m_1)$, where $A(x) \stackrel{\text{def}}{=} \bar{c}\,x.\underline{0} \parallel \bar{c}\,m_2.A(m_3)$. Note that:

$$
\begin{aligned}
I(A(m_3), \{A\}) &= m_3 \\
I(\bar{c}\,x.\underline{0}, \{A\}) &= I(\underline{0}, \{A\}) = \emptyset
\end{aligned}
$$

$$\begin{array}{lll} I(\bar{c}\,m_2.A(m_3),\{A\}) & = \{m_2\} \cup I(A(m_3),\{A\}) & = \{m_2,m_3\} \\ I(A(x),\{A\}) & = I(\bar{c}\,x.\underline{0},\{A\}) \cup I(\bar{c}\,m_2.A(m_3),\{A\}) & = \{m_2,m_3\} \\ I(A(m_1),\emptyset) & = \{m_1\} \cup I(A(x),\{A\}) & = \{m_1,m_2,m_3\} \end{array}$$

Thus, we have $ID(A(m_1)) = I(A(m_1),\emptyset) = \{m_1,m_2,m_3\}$.

Now, let $\phi_I \subseteq \mathscr{M}$ be the initial knowledge that we would like to give to the intruders, 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 $\mathscr{E}_C^{\phi_I}$ of enemies as $\mathscr{E}_C^{\phi_I} = \{X \mid sort(X) \subseteq C \text{ and } ID(X) \subseteq \mathscr{D}(\phi_I)\}$.

To see how $\mathscr{E}_C^{\phi_I}$ prevents the problem presented in Example 2.2, consider again the enemy $X(m_X,k_{AB})$ of that example. To specify that k_{AB} is secret, it is now sufficient to require that $k_{AB} \notin \mathscr{D}(\phi_I)$. Since $ID(X(m_X,k_{AB})) = \{m_X,k_{AB}\}$, we finally have that $X(m_X,k_{AB}) \notin \mathscr{E}_C^{\phi_I}$.

2.4. Semantic equivalences

In this section we define some semantic equivalences that we will use to formalize security properties. Each equivalence is based on a particular notion of *indistinguishable behavior* which we informally describe in the following:

- *Trace equivalence* requires, for two processes to be equivalent, that the set of their possible execution sequences is exactly the same. This must be true even when the processes are exposed to every possible intruder. Such a definition would be too strong for our purposes if used without restricting public channels: as a matter of fact, a generic intruder would be able to guess secret values hence breaking cryptography. Thus, we will always use trace equivalence inside definitions that restrict the scope of public channels. Trace equivalence is used in Section 4 for the definition of NDC and in Section 5 for the definition of Agreement.
- *May-testing equivalence* requires that equivalent processes cannot be distinguished by any process (tester) that does not know the secret values. This definition incorporates the fact that the tester is not able to break cryptography and is thus suitable to work on "open" processes. May-Testing is used in Section 3 for the definition of spi-calculus authentication.

We will also define some technical notions which will be useful in proofs:

- *Barbed pre-congruence* relation, denoted with $P \lesssim P'$, requires that the "greater" process P' is able to simulate step-by-step the "smaller" one P, in any testing context. This notion is useful to prove that two processes are testing equivalent.
- A *structural equivalence* is also defined, in order to simplify the manipulation of parallel and restriction operators in the proofs.

We now formally define the notions described above.

2.4.1. 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 show exactly the same execution sequences (called *traces*). We need a transition relation $E \stackrel{a}{\Rightarrow} E'$ which does not consider internal τ moves. It is a shorthand for $E(\stackrel{\tau}{\rightarrow})^* E_1 \stackrel{a}{\rightarrow} E_2(\stackrel{\tau}{\rightarrow})^* E'$. For a trace $\gamma = a_1 \ldots a_n$ we write $E \stackrel{\gamma}{\Rightarrow} E'$ if $E \stackrel{a_1}{\Rightarrow} E_1 \stackrel{a_2}{\Rightarrow} \cdots \stackrel{a_{n-1}}{\Rightarrow} E_{n-1} \stackrel{a_n}{\Rightarrow} E'$ for some $E_1, \ldots, E_{n-1}$. Note that $E \Rightarrow E'$ stands for a possibly empty sequence of internal transitions, i.e., $E(\stackrel{\tau}{\rightarrow})^* E'$. The set $Tr(E)$ of *traces associated with* E is then defined as $Tr(E) = \{\gamma \in (Act \setminus \{\tau\})^* \mid \exists E' : E \stackrel{\gamma}{\Rightarrow} E'\}$.

Definition 2.4. Let $E, F \in \mathscr{E}$. We write $E \leqslant_{\text{trace}} F$ iff $Tr(E) \subseteq Tr(F)$. We also say that E and F are *trace equivalent* (notation $E \approx_{\text{trace}} F$) iff $E \leqslant_{\text{trace}} F$ and $F \leqslant_{\text{trace}} E$.

2.4.2. Testing equivalence for CryptoSPA

In this section we give a notion of behavioral 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 a weaker version of classical testing equivalence, which requires that equivalent processes should be indistinguishable by *any* process [5].

We define the notion of experiment as given in the spi-calculus. A *test* is a pair (T, β), where T is a process called tester and β is a *barb*, i.e., a channel c or $\bar{c}$. A process P *exhibits a barb* β (denoted by $P \downarrow \beta$) iff for some message m and process P', we have $P \xrightarrow{\beta m} P'$. Moreover P *converges on a barb* β (denoted by $P \Downarrow \beta$) iff $P(\stackrel{\tau}{\rightarrow})^* P'$ and $P' \downarrow \beta$. Now, we say that a process P *immediately passes a test* (T, β) iff $(P \parallel T) \downarrow \beta$. We also say that a process P *passes a test* (T, β) iff $(P \parallel T) \Downarrow \beta$.

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: $\mathscr{E}^{\phi_I} = \{X \mid X \in \mathscr{E} \text{ and } ID(X) \subseteq \mathscr{D}(\phi_I)\}$. Note that this set is strictly larger than the set of enemies $\mathscr{E}_C^{\phi_I}$ defined in the previous section. The reason is that such enemies can only communicate over the public channels in C.

Definition 2.5. For all $P, Q \in \mathscr{E}$, we write $P \leqslant_{\text{may}} Q$ iff $\forall T \in \mathscr{E}^{\phi_I}$, $\forall \beta \in I \cup O : P \parallel T \Downarrow \beta$ *implies* $Q \parallel T \Downarrow \beta$. We also say that P is *may-testing equivalent* to Q (notation $P \approx_{\text{may}} Q$) iff $P \leqslant_{\text{may}} Q$ and $Q \leqslant_{\text{may}} P$.

It is easy to prove that $\approx_{\text{may}}$ is an equivalence relation on CryptSPA processes.

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 enemy. In fact, a tester 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 together with every possible enemy.

Our notion of may-testing equivalence differs from the definition given for the spi-calculus, where every possible spi-calculus process can be a test. This difference is due to the way secret messages are dealt with. The restriction operator of the

(s)pi-calculus supports the specification of *new*, and 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. [3]

Proposition 2.6. *Let $P,Q\in\mathscr{E}$. Then, $P\leqslant_{\text{trace}} Q$ implies $P\leqslant_{\text{may}} Q$.*

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 $A(x_1,x_2)\stackrel{\text{def}}{=}[\langle x_1,x_2\rangle\vdash_{\text{enc}} x]\bar{c}x$ that encrypts x_1 with x_2 and sends it out on channel c. If k_A is secret (and also all the messages encrypted with k_A), then $A(m_1,k_A)\approx_{\text{may}} A(m_2,k_A)$ even if $m_1\neq m_2$ and m_1,m_2 are public. On the other hand, if $m_1\neq m_2$ then we have $A(m_1,k_A)\not\approx_{\text{trace}} A(m_2,k_A)$ because $\bar{c}\{m_1\}_{k_A}$ is a trace for $A(m_1,k_A)$ but not for $A(m_2,k_A)$.

We can also prove that our may-testing preorder is preserved by the parallel composition with testers and by the restriction operator.

Proposition 2.7. *Let $P,Q\in\mathscr{E}$ be two processes, $R\in\mathscr{E}^{\phi_I}$ a tester and L a set of channels. If $P\leqslant_{\text{may}} Q$ then*
(i) $P\parallel R\leqslant_{\text{may}} Q\parallel R$;
(ii) $P\backslash L\leqslant_{\text{may}} Q\backslash L$.

Corollary 2.8. *Let $P,Q\in\mathscr{E}$. If $P\approx_{\text{may}} Q$ then for all $R\in\mathscr{E}^{\phi_I}$ we have $P\parallel R\approx_{\text{may}} Q\parallel R$ and $P\backslash L\approx_{\text{may}} Q\backslash L$.*

We will exploit these results when comparing the two different notions of authentication we are going to introduce in the following sections.

2.4.3. Barbed bisimulation

Barbed bisimulation [25] provides very efficient proof techniques for verifying the other equivalence notions defined so far.

Definition 2.9. A relation $\mathscr{S}\subseteq\mathscr{E}\times\mathscr{E}$ is a barbed simulation if $(P,Q)\in\mathscr{S}$ implies
- if $P\downarrow\beta$ then $Q\downarrow\beta$,
- if $P\stackrel{\tau}{\rightarrow}P'$ then there exists Q' such that $Q\stackrel{\tau}{\rightarrow}Q'$ and $(P',Q')\in\mathscr{S}$.

A barbed bisimulation is a symmetric relation $\mathscr{S}$ such that both $\mathscr{S}$ and $\mathscr{S}^{-1}$ are barbed simulations. The union of all barbed simulations is represented by $\trianglelefteq$.

[3] It is worthwhile noticing that in [22,20] a language similar to CryptoSPA is equipped with a form of secret generation.

Definition 2.10. A relation $\mathscr{S} \subseteq \mathscr{E} \times \mathscr{E}$ is a barbed pre-congruence (w.r.t. ϕ_I) iff for $(P,Q) \in \mathscr{S}$ and for all $R \in \mathscr{E}^{\phi_I}$ we have $P \| R \trianglelefteq Q \| R$.

We define $\precsim$ to be the largest barbed pre-congruence. As usual we can define the notion of barbed equivalence, and the weak versions of these preorders and equivalences. In particular, a barbed weak simulation (denoted with $\trianglelefteqq$) is a relation defined as in Definition 2.9 by simply replacing $Q \downarrow \beta$ with $Q \Downarrow \beta$ and $Q \xrightarrow{\tau} Q'$ with $Q(\xrightarrow{\tau})^* Q'$. A barbed weak pre-congruence (denoted with $\lessapprox$) is defined as in Definition 2.10 by replacing $\trianglelefteq$ with $\trianglelefteqq$.

The following result is useful to prove that two processes are related by $\leqslant_{\text{may}}$.

Proposition 2.11. $\precsim$, $\lessapprox \subseteq \leqslant_{\text{may}}$.

We also define *structural equivalence* as follows.

Definition 2.12. Let $P,Q,R \in \mathscr{E}$ and $L,L_1 \subseteq I$. Then, we define $\equiv$ as the least equivalence relation closed under the following rules:
(1) $P \| \underline{0} \equiv P$;
(2) $P \| Q \equiv Q \| P$;
(3) $P \|(Q \| R) \equiv (P \| Q) \| R$;
(4) $\underline{0}\backslash L \equiv \underline{0}$;
(5) $P\backslash L\backslash L_1 \equiv P\backslash L_1\backslash L$;
(6) $(P \| Q)\backslash L \equiv P\backslash L \| Q$ if $sort(Q) \cap L = \emptyset$.

Structural equivalence will be useful to manipulate parallel and restriction operators in the proofs. Moreover, the following lemma shows that structural equivalence is a barbed simulation; hence, in the following, it will be sound to use barbed simulation up to "equivalent processes" as a proof technique.

Lemma 2.13. *Let $P,Q \in \mathscr{E}$. Then, $P \equiv Q$ implies*
- *if $P \downarrow \beta$ then $Q \downarrow \beta$,*
- *if $Q \xrightarrow{\tau} Q'$ then $P \xrightarrow{\tau} P'$ with $P' \equiv Q'$.*

Definition 2.14. A relation $\mathscr{S} \subseteq \mathscr{E} \times \mathscr{E}$ is a barbed simulation up to $\equiv$ if $(P,Q) \in \mathscr{S}$ implies
- if $P \downarrow \beta$ then $Q \downarrow \beta$,
- if $P \xrightarrow{\tau} P'$ then there exists Q' such that $Q \xrightarrow{\tau} Q'$ and $P' \equiv \mathscr{S} \equiv Q'$,

where $P' \equiv \mathscr{S} \equiv Q'$ means that for some $P'' \equiv P'$ and $Q'' \equiv Q'$ we have $(P'',Q'') \in \mathscr{S}$.

Proposition 2.15. *If $\mathscr{S}$ is a barbed simulation up to $\equiv$ then $\equiv \mathscr{S} \equiv$ is a barbed simulation.*

An analogous definition and result can be given for *barbed weak simulation* up to $\equiv$.

3. Message authentication in the spi-calculus

In [1,2] an interesting notion of authentication is proposed. Consider a process $S(\mathbf{m})$ representing a protocol[4] which has the aim of transmitting message $\mathbf{m}$ from a party A to a party B. $S(\mathbf{m})$ guarantees the authentication of message $\mathbf{m}$ if whenever a message m is delivered to B by $S(\mathbf{m})$, then m must be the same message as $\mathbf{m}$.

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

In [1,2] a general method for generating the specification $S_{\text{spec}}(\mathbf{m})$ is not provided. Instead, the construction of $S_{\text{spec}}(\mathbf{m})$ is illustrated by several example protocols. Intuitively, it can be obtained as follows: in the protocol $S(\mathbf{m})$, every time B accepts a message m, such a message is replaced by $\mathbf{m}$. In the following we formalize this intuition obtaining a general method for the construction of $S_{\text{spec}}(\mathbf{m})$ starting from $S(\mathbf{m})$. We feel that our formalization is general enough to model the significant examples of cryptographic protocols reported in [1,2] (see also Section 6.1.1).

Note 1. As in [1,2], we assume that $S(\mathbf{m})$ 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. Here, we assume that the execution part can only send messages on public channels $c \in C$, while the continuation, by definition, does not take part to 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 example of a protocol specification in this style.

Example 3.1. We specify here a trivial protocol where Alice sends to Bob a message $\mathbf{m}$ as plain text over an insecure channel c:

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

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

$$A(x) \stackrel{\text{def}}{=} \bar{c}\,x, \quad B_{\text{spec}}(t) \stackrel{\text{def}}{=} c(y).[y = t]F(t), \quad S_{\text{spec}}(x) \stackrel{\text{def}}{=} A(x) \,\|\, B_{\text{spec}}(x)$$

where it is possible to give to Bob a message t that is "magically" checked against y before activating the continuation F. In $S_{\text{spec}}(\mathbf{m})$ we have that Bob can only accept the correct message $\mathbf{m}$, then behaving like $F(\mathbf{m})$. This represents the secure version of

[4] In the following, we will often use the word "protocol" instead of "process".

$S(\mathbf{m})$, i.e., a version where the received message is always authentic by construction. Clearly, $S(\mathbf{m})$ does not guarantee any authentication of $\mathbf{m}$: any external user can introduce a fake message $\mathbf{m}'$ on c that will be accepted by Bob. As a consequence, $S(\mathbf{m})$ can move to $F(\mathbf{m}')$ while $S_{\text{spec}}(\mathbf{m})$ cannot.

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

$$A'(x) \stackrel{\text{def}}{=} \bar{c}\,x, \quad B' \stackrel{\text{def}}{=} c(y).\bar{p}\,y, \quad S'(x) \stackrel{\text{def}}{=} (A'(x) \parallel B' \parallel p(z).F(z)) \backslash \{p\}$$

As a matter of fact, $A'(x) \parallel B'$ 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 special form $S'(x)$ for the protocol is trace and may-testing equivalent to the initial one $S(x)$, provided that $p \notin sort(F)$.

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 protocol S, we denote all of its occurrences of continuations[5] as $\{F_1(x_1), \ldots, F_n(x_n)\}$, where x_i represents the only free variable of F_i. From S we derive a process $S_{nf}(m_1, \ldots, m_n)$ in normal form as follows:

$$\left(S' \parallel \prod_{i \in 1,\ldots,n} p_{F_i}(x_i).F_i(x_i) \right) \backslash \vec{p} \tag{1}$$

where S' is the process S where every continuation $F_i(x_i)$ is replaced by $\overline{p_{F_i}}(x_i)$, and $\vec{p} = \{p_{F_1}, \ldots, p_{F_n}\}$ 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 $\vec{p}$ are indexed with the continuations F_i. This is useful for managing multiple concurrent sessions between senders and receivers which can be modeled by considering n copies of the sender and n copies of the receiver in parallel. We assume that syntactically equal continuations (up to renaming of bound variables, i.e., α-conversion) correspond to the same protocol between two users but in different parallel sessions.

Given this particular form for protocols, it is quite natural to derive a secure specification. More precisely, given the normal form $S_{nf}(m_1, \ldots, m_n)$ as in (1), it is sufficient to define $S_{\text{spec}}(m_1, \ldots, m_n)$ as follows:

$$\left(S' \parallel \prod_{i \in 1,\ldots,n} p_{F_i}(x_i).[x_i = m_i]F_i(m_i) \right) \backslash \vec{p}. \tag{2}$$

Note that every continuation is enabled only if the received message x_i is equal to the correct message m_i. Note also that in the case of multiple sessions, this simply requires that a "correct" multiset of messages is delivered from one process to another one, in whatever possible order (see Section 6.1.1 for an example).

[5] We assume that continuations are not dynamically duplicated in the protocol.

Remark 3.2. In the following, we will always consider protocols in the normal form (1) and specifications in the form (2).

As done in the spi-calculus, we will always assume that the messages $\mathbf{m}_i$ to be delivered by the protocol are not secret, otherwise it would not be possible to observe them with testing equivalence.

Assumption 1. When we consider a message $\mathbf{m}$ to be delivered by a protocol we always assume that $\mathbf{m} \in \mathscr{D}(\phi_I)$.

Another important assumption (common to the spi-calculus approach) about the continuations is that they should not know the protocol secrets. Otherwise, they could help the tester attacking the protocol by making such secrets public.

Assumption 2. For all continuations $F(x)$ and for all $m \in \mathscr{D}(\phi_I)$ we have $F(m) \in \mathscr{E}^{\phi_I}$.

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

Definition 3.3. A protocol S guarantees weak spi-authentication iff for all vectors $(\mathbf{m}_1, \ldots, \mathbf{m}_n)$ of messages: $S(\mathbf{m}_1, \ldots, \mathbf{m}_n) \approx_{\text{may}} S_{\text{spec}}(\mathbf{m}_1, \ldots, \mathbf{m}_n)$.

The following lemma states that the behavior of a specification may always be simulated by the protocol. So, as expected, the specification only shows a (correct) subset of the possible behaviors of the protocol.

Lemma 3.4. *For all vector of messages* $\vec{\mathbf{m}} = (\mathbf{m}_i)_{i \in 1 \ldots n}$, $S_{\text{spec}}(\vec{\mathbf{m}}) \lesssim S(\vec{\mathbf{m}})$.

As a consequence, we have that *spi*-authentication concerns whether the specification is an upper bound of the protocol.

Corollary 3.5. *A protocol S guarantees weak spi-authentication iff for all vectors* $(\mathbf{m}_1, \ldots, \mathbf{m}_n)$ *of messages*: $S(\mathbf{m}_1, \ldots, \mathbf{m}_n) \lesssim S_{\text{spec}}(\mathbf{m}_1, \ldots, \mathbf{m}_n)$.

In [1], another notion of authentication is proposed which includes a quantification over the continuations F_i. In this way, it is 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 3.6. Consider again the simple protocol of Example 3.1. Now, consider the special case where $F(x) = \underline{0}$. We obtain that for every $\mathbf{m}$, $S(\mathbf{m})$ is equivalent to $S_{\text{spec}}(\mathbf{m})$. Indeed, $S(\mathbf{m})$ and $S_{\text{spec}}(\mathbf{m})$ become the same process if $F(x) = \underline{0}$. However,

we have already observed that this trivial protocol cannot guarantee the authenticity of $\mathbf{m}$. 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 behavior of a continuation does not depend on the message x then it cannot help detecting authentication attacks.

Let $\vec{F}$ be a vector of continuations $(F_1,\ldots,F_n)$. We will denote with $S^{\vec{F}}$ the protocol S where its continuations are replaced by $\vec{F}$. Moreover, the initial vector of a protocol S will be denoted by $\vec{F}^*$.

When we quantify over all possible vectors of continuations, we require that equal continuations (up to α-conversion) are always replaced by equal continuations. More formally, we say that two vectors $(F_1,\ldots F_n),(F'_1,\ldots,F'_n)$ are *compatible* if and only if $\forall i,j\in\{1,\ldots n\}$: $F_i=F_j$ iff $F'_i=F'_j$. We then consider only the vectors of continuations that are compatible with the initial one $\vec{F}^*$. This condition is necessary if we want to analyze multiple sessions where many instances (all with the same continuation) of a process are considered (see Section 6.1.1). Hence, we obtain the following more general definition:

Definition 3.7. A protocol S guarantees (strong) spi-authentication iff for all the vectors of continuations $\vec{F}$ *compatible* with $\vec{F}^*$, $S^{\vec{F}}$ guarantees weak spi-authentication.

In the following, with spi-authentication we will always refer to the strong spi-authentication property.

Example 3.8. In order to illustrate the definition above, we consider once more the simple insecure protocol of Example 3.1. We have already shown that it does not guarantee any authentication of $\mathbf{m}$. Here we formally prove that it does not guarantee spi-authentication.

To this purpose, it is enough to consider the continuation $\vec{F}=(\overline{out}\,z)$, the message $\vec{\mathbf{m}}=(\mathbf{m})$ and the tester (T,β) with $T=\bar{c}\,\mathbf{m}'.out(x).[x=\mathbf{m}']\bar{\beta}\,x$ and $\mathbf{m}'\neq\mathbf{m}$. Now it is easy to see that $S^{\vec{F}}(\mathbf{m})\not\approx_{\text{may}} S^{\vec{F}}_{\text{spec}}(\mathbf{m})$. As a matter of fact,

$$
\begin{aligned}
S^{\vec{F}}(\mathbf{m}) \parallel T &= \bar{c}\,\mathbf{m} \parallel c(y).\overline{out}\,y \parallel \bar{c}\,\mathbf{m}'.out(x).[x=\mathbf{m}']\bar{\beta}\,x\\
&\xrightarrow{\tau} \bar{c}\,\mathbf{m} \parallel \overline{out}\,\mathbf{m}' \parallel out(x).[x=\mathbf{m}']\bar{\beta}\,x\\
&\xrightarrow{\tau} \bar{c}\,\mathbf{m} \parallel \underline{0} \parallel \bar{\beta}\,x \downarrow \beta
\end{aligned}
$$

hence, $S^{\vec{F}}(\mathbf{m}) \parallel T \Downarrow \beta$. On the other hand, whatever execution sequence is performed by process $S^{\vec{F}}_{\text{spec}}(\mathbf{m}) \parallel T$, the only $\overline{out}\,y$ action that can be executed by $S^{\vec{F}}_{\text{spec}}(\mathbf{m})$ is $\overline{out}\,\mathbf{m}$. So T will always receive $\mathbf{m}$ which is different from $\mathbf{m}'$ and the test $[x=\mathbf{m}']$ will never be passed. So, $S^{\vec{F}}_{\text{spec}}(\mathbf{m}) \parallel T \not\Downarrow \beta$

In the following, we will show that there exists a vector of *canonical continuations* $\vec{F}'$ such that the weak spi-authentication of $S^{\vec{F}'}$ implies the spi-authentication

of S. Thus, these special continuations allow us to avoid the universal quantification over all possible vectors $\vec{F}$. In particular, we consider the special vector $\vec{F}'$ where $F_i'(x_i) = \overline{out_{F_i^*}}(x_i)$ and such that the channels $out_{F_i^*}$ are private, i.e., not in C. Note that this vector is compatible with $\vec{F}^*$. We have the following result.

Proposition 3.9. *S guarantees spi-authentication iff $S^{\vec{F}'}$ guarantees weak spi-authentication, with $F_i'(x_i) = \overline{out_{F_i^*}}(x_i)$.*

Intuitively, this interesting result states that there exists a *canonical continuation* $\vec{F}'$ that can be considered in order to prove spi-authentication for every possible continuation. We will use this result to compare such a notion of authentication with the NDC-based one (see Section 6.1).

The solution of using may-testing equivalence for authentication verification is very elegant, but mixing the tester and the intruder could generate some confusion. 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 is related to the particular form of the protocols. 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) [15]. NI was proposed in system security as a model for the detection of all possible interferences of a certain group of users with another one. It has been formalized in different ways (see, e.g., [9,15,24,28]), and also applied in the verification of various properties of security protocols [6–8], with authentication among them. 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 behaves exactly in the same way with or without the enemy. 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 [9] and called *non-deducibility on compositions* (*NDC*). The idea is the following: the group U of *untrusted* users that must not interfere with the other users (in trusted group $\mathscr{T}$) is characterized by the set of actions that its components can execute. Public channels in C are the channels which processes in U can use for communication. A process S is NDC if every possible process $X \in U$ composed with S is not able to modify the behavior of S observed from the point of view of the users in $\mathscr{T}$. Thus, U corresponds to $\mathscr{E}_C^{\phi_I}$ and NDC can be defined as follows, where we always consider processes S which are in (normal) form (1).

Definition 4.1. A process S is NDC iff $\forall X \in \mathscr{E}_C^{\phi_I}\ (S \,\|\, X)\backslash C \approx_{\text{trace}} S\backslash C$.

The only difference with respect to the definition given in the original model (SPA, with no cryptography) is that the knowledge of processes $X \in U$ is bounded by ϕ_I. In CryptoSPA, this is required to correctly model secret values, as observed in Section 2.3. NDC requires that untrusted processes in $\mathscr{E}_C^{\phi_I}$ are not able to change the behavior of the process observed from processes in $\mathscr{T}$ and represented by $S\backslash C$. As a matter of fact, $S\backslash C$ is the process where no untrusted activity is allowed. If it is equivalent to $(S \,\|\, X)\backslash C$, this means that X is not able to modify in any observable 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 4.2. A process S is TNDC iff $\forall X \in \mathscr{E}_C^{\phi_I}\ (S \,\|\, X)\backslash C \approx_{\text{may}} S\backslash C$.

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

Definition 4.3. Let S be a protocol. S guarantees $(T)NDC$-authentication iff for all messages $\mathbf{m}_1, \ldots, \mathbf{m}_n$ we have that $S(\mathbf{m}_1, \ldots, \mathbf{m}_n) \in (T)NDC$.

Note that in the definition above we do not take care of continuations. In particular, the fact that $S(\mathbf{m}_1, \ldots, \mathbf{m}_n) \in (T)NDC$ requires that $S(\mathbf{m}_1, \ldots, \mathbf{m}_n)$ composed with whatever enemy $X \in \mathscr{E}_C^{\phi_I}$ is equivalent to $S(\mathbf{m}_1, \ldots, \mathbf{m}_n)\backslash C$. Since C represents the set of channels over which the parties communicate, then $S(\mathbf{m}_1, \ldots, \mathbf{m}_n)\backslash C$ corresponds to a secure specification (execution with protected channels). In this approach we obtain, in a sense, the secure specification for free.

Example 4.4. Similarly to what we did in Example 3.8, we apply the NDC definition above to formally prove that the protocol of Example 3.1 is insecure, with respect to authentication of $\mathbf{m}$.

It is interesting to see that since we restrict the behavior of the process on the channels in C we actually do not observe directly the communication, but only the behavior on the continuations. We obtain a specification that is somehow simpler than the one needed in the spi-calculus authentication approach. As a matter of fact, $S(\mathbf{m})\backslash C$ is trace and may-testing equivalent to the process $F(\mathbf{m})$, which is just the expected (correct) continuation for the protocol.

Proving that the protocol is not NDC is rather simple. Consider a message $\mathbf{m}$ and an attacker $X = \bar{c}\,\mathbf{m}'$ with $\mathbf{m}' \neq \mathbf{m}$. Now it is easy to see that $(S(\mathbf{m}) \,\|\, X)\backslash C \not\approx_{\text{trace}} S(\mathbf{m})\backslash C$ when $F(\mathbf{m}') \not\leqslant_{\text{trace}} F(\mathbf{m})$. As a matter of fact, it quite easy to prove that $(S(\mathbf{m}) \,\|\, X)\backslash C \approx_{\text{trace}} F(\mathbf{m}) + F(\mathbf{m}') \not\leqslant_{\text{trace}} F(\mathbf{m}) \approx_{\text{trace}} S(\mathbf{m})\backslash C$ when $F(\mathbf{m}') \not\leqslant_{\text{trace}} F(\mathbf{m})$. The last condition is rather intuitive and can be read as "the fact that Bob receives a message different from the expected one has some observable effect on his future behavior". If this is not true, the value of the received message is not relevant and checking the authentication of $\mathbf{m}$ becomes useless.

5. The agreement property

In this section we recall the notion of *Agreement* [18] as formalized in the general schema of [14]. The basic idea of the *Agreement* property is the following. A protocol guarantees to a responder B *Agreement* with an initiator A on a set of data items ds if, whenever B (acting as a responder) completes a run of the protocol, apparently with the initiator A, then:

- A has previously been running the protocol, apparently with B, and A was acting as initiator in this run,
- A and B agreed on the data values corresponding to all the variables in ds,
- each run of B corresponds to a unique run of A.

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

As done in [18], we assume that the actions representing the running and the commit are correctly specified in the protocol. We can see them as output actions over two particular channels *running_ini* and *commit_res*. For simplicity, in the examples we only analyze the case where A is the initiator and B is the responder, and the set ds of data items is composed only by a single datum d from a set D. However, the specification can be easily extended in order to cover all the cases studied in [18]. Let $NotObs(P) = sort(P)\backslash(C \cup \{running\_ini, commit\_res\})$ be the set of channels in P that are not public and are different from *running_ini* and *commit_res*, i.e., that will not be observed. We can then define function α_{Agree} as follows:

$$P'(x,y) = \sum_{d\in D} \overline{running\_ini}\,(x,y,d).\overline{commit\_res}\,(y,x,d)$$

$$P'' = \sum_{c\in NotObs(P)} c(x).P'' + \sum_{\substack{c\in NotObs(P)\\ m\in Msg(c)}} \bar{c}\,m.P''$$

$$\alpha_{\mathrm{Agree}}(P) = P'' \parallel P'(A,B)$$

Note that P'' is essentially the process that executes every possible action over channels in $sort(P)$ which are not in C and are different from *running_ini* and *commit_res*.[6] Given P, $\alpha_{\mathrm{Agree}}(P)$ represents the most general process that satisfies the *Agreement* property and has the same sort as P. As a matter of fact in $\alpha_{\mathrm{Agree}}(P)$ action $\overline{running\_ini}$

[6] Note also that inputs are ignored, as in P'' there is no occurrence of a free x. The reason is that P'' already knows all the messages can it can output.

(A,B,d) always precedes $\overline{commit\_res}\,(B,A,d)$ for every datum d, and every combination of the other actions of P can be executed. In order to analyze more than one session, it is sufficient to consider an extended α which has several processes $P'(A,B)$ in parallel. For example, for n sessions we can consider the following:

$$\alpha_{\mathrm{Agree}}(P) = P'' \parallel \underbrace{P'(A,B) \parallel \cdots \parallel P'(A,B)}_{n}$$

We want that even in the presence of an enemy, P does not execute traces that are not in $\alpha_{\mathrm{Agree}}(P)$. So we can give the following definition:

Definition 5.1. $P \in \mathscr{E}$ satisfies Agreement iff $\forall X \in \mathscr{E}_C^{\phi_I}$: $(P \parallel X)\backslash C \leqslant_{\mathrm{trace}} \alpha_{\mathrm{Agree}}(P)$

Note that in [18] it is only required that *Agreement* holds when the process is composed with a particular intruder, which should intuitively be the most powerful one. In [14] we have formally proved that this simpler requirement is sufficient (and necessary) to guarantee our version of *Agreement*. This point is discussed further in Section 7.

Example 5.2. We consider here a simple example to illustrate the *Agreement* property. One of the aim of *Agreement* is to verify whether a protocol guarantees *entity authentication* of one party with respect to another one. As an example, consider the following (flawed) protocol: Alice wants to authenticate herself to a server S; in order to achieve this, she sends her login "A" and password "wonderland", both encrypted with a key which is shared with the server S. The encryption should avoid the discovering of Alice's password by a malicious third party.

We specify this protocol as follows (we directly enrich the protocol with *running_ini* and *commit_res* actions):

$$A = \overline{running\_ini}\,(A,S).\ \bar{c}\{A, wonderland\}_{k_{AS}}$$

$$\begin{aligned} S = {} & c(x).(\\ & [\langle x, k_{AS}\rangle \vdash_{\mathrm{dec}} y]\ [y \vdash_{\mathrm{fst}} l]\ [y \vdash_{\mathrm{snd}} p] \\ & [l = A]\ [p = wonderland]\ \overline{commit\_res}\,(S,A) \\ & + [\langle x, k_{BS}\rangle \vdash_{\mathrm{dec}} y] \ldots (\text{for authenticating another user "}B\text{"}) \\ & \cdots \\ & + [\langle x, k_{US}\rangle \vdash_{\mathrm{dec}} y] \ldots (\text{other possible users}) \end{aligned}$$

$$P = A \parallel S \parallel \cdots \parallel S$$

where $k_{AS}, wonderland \notin \phi_I$, as they are secret. We suppose to have more than one server ready to verify authentication requests from the users. An important property that

this protocol should guarantee is that no malicious party should be able to impersonate any users with respect to the server S. We now show that this protocol is insecure by proving that it does not satisfy *Agreement*. In particular, it is sufficient to consider the following attacker (belonging to $\mathscr{E}_C^{\phi_I}$) which intercepts the message from Alice and replays it later: $X = c(x).\bar{c}x.\bar{c}x$. Now it is easy to see that $(P \| X)\backslash C \not\leq_{\text{trace}} \alpha_{\text{Agree}}(P)$ by considering the following execution:

$$
\begin{array}{lll}
(P \| X)\backslash C & = & (A \| S \| \cdots \| S \| X)\backslash C \\
& \xrightarrow{\overline{running\_ini}\,(A,S)} & (\underline{0} \| S \| \cdots \| S \| \bar{c}\{A, wonderland\}_{k_{AS}} \\
& & \bar{c}\{A, wonderland\}_{k_{AS}})\backslash C \\
& \Rightarrow & (\underline{0} \| \overline{commit\_res}\,(S,A) \| \overline{commit\_res}\,(S,A) \| \\
& & \| S \cdots \| S \| \underline{0})\backslash C \\
& \xrightarrow{\overline{commit\_res}\,(S,A),\ \overline{commit\_res}\,(S,A)} & \\
& \Rightarrow & (\underline{0} \| \underline{0} \| \underline{0} \| S \cdots \| S \| \underline{0})\backslash C
\end{array}
$$

the corresponding trace is

$$\overline{running\_ini}\,(A,S).\overline{commit\_res}\,(S,A).\overline{commit\_res}\,(S,A)$$

where a commit with no previous running is present and which, consequently, is not a trace for $\alpha_{\text{Agree}}(P)$. The enemy is able to impersonate Alice by just replaying the same encrypted message. Note that the enemy is not learning the password as he does not know that key K_{AS}. Nevertheless, he is able to carry out the attack.

6. Comparison

In this section, we formally compare the notions of authentication presented in the previous sections. Specifically, we compare the notion of authentication proposed for the spi-calculus both with the TNDC-based authentication and with the *Agreement* one.

6.1. Spi-calculus and NDC-based authentication

We have already pointed out some of the similarities between spi-calculus and NDC-based authentication: both properties are based on a notion of behavioral equivalence; moreover, they both check whether the "process 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 process is implicitly checked against all the possible interactions with the (hostile) environment through the use of the may-testing equivalence. There, the tester plays simultaneously both the role of the attacker and the role of the observer (see Fig. 4). On the other hand, the TNDC-based approach performs an explicit quantification over all possible intruders, then observing the outcome of the attack (see Fig. 5). 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

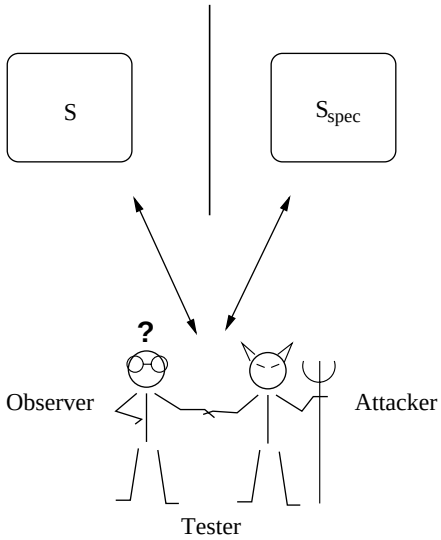

Fig. 4. Testers as "observers and attackers" in spi-authentication.

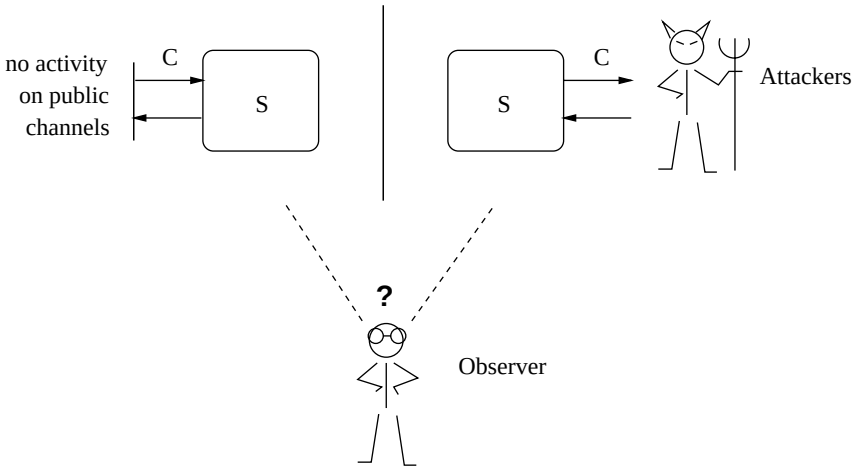

Fig. 5. Separated attackers and observers in NDC-authentication.

TNDC-authentication. (Later on, we will prove that also the converse holds, under some mild assumptions.)

Proposition 6.1. *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 guarantees TNDC-authentication.*

Proof. We have to prove that

$$\forall X \in \mathscr{E}_C^{\phi_I} \quad (S(\vec{\mathbf{m}}) \,\|\, X)\backslash C \approx_{\text{may}} S(\vec{\mathbf{m}})\backslash C \tag{3}$$

for all vectors of messages $\vec{\mathbf{m}}$. Since $S(\vec{\mathbf{m}}) \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})$ and $\approx_{\text{may}}$ is a congruence w.r.t. the restriction operator and the parallel composition with processes in $\mathscr{E}^{\phi_I}$ (see Corollary 2.8), then proving (3) is equivalent to proving the following:

$$\forall X \in \mathscr{E}_C^{\phi_I} \quad (S_{\text{spec}}(\vec{\mathbf{m}}) \,\|\, X)\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$$

which is true by the hypothesis $S_{\text{spec}}(\vec{\mathbf{m}}) \in TNDC$. □

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

WFC1 For every vector of messages $\vec{\mathbf{m}}$, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k\in 1..n} F_k(\mathbf{m}_k)$.

This condition is a very natural one as it requires that all the continuations of the specifications when there is no attacker at all, are eventually enabled. In other words, the specification is well formed for not containing unreachable continuations. If it does, then some useless redundancy is present in S_{spec}. Under this condition we can prove the following lemma.

Lemma 6.2. *Assume that for every vector of messages* $\vec{\mathbf{m}}$, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k\in 1..n} F_k(\mathbf{m}_k)$. *Then,* S_{spec} *guarantees TNDC-authentication.*

This lemma gives a formal evidence of the correctness of the way we define specifications S_{spec}. As a consequence of this lemma we obtain that, in general, spi-authentication is at least as discriminating as the TNDC-based one.

Corollary 6.3. *Let S be a protocol such that for every* $\vec{\mathbf{m}}$, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k\in 1..n} F_k(\mathbf{m}_k)$. *If S guarantees spi-authentication then S guarantees TNDC-authentication.*

Proof. The proof directly follows from Proposition 6.1 and Lemma 6.2. □

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 (see also Fig. 5), we have to be sure that this "splitting" can be performed without loosing discriminating power. For example, as our intruders are not able to communicate over the channels of the continuations, one may think, at a first glance, that TNDC may miss some possible attack. We show that this is not the case.

We need to precisely define when the delivering of a set of messages to the continuations is correct in a protocol. Intuitively, it is correct if the same delivering is possible also in the specification. To formalize this concept we need to define the *vectors of activations*.

Definition 6.4. Let $S^{\vec{F}}(\vec{\mathbf{m}})$ be a protocol and T be a tester and consider a computation γ between them. The vector of activations of γ (denoted with $Activations(\gamma)$) is the vector of all the activated continuations with the received message:

$$Activations(\gamma) = (F_k(\mathbf{m}') \mid p_{F_i}(x_i).F_i(x_i) \text{ synchronizes with } \bar{p}_{F_k}\mathbf{m}' \text{ in } \gamma)$$

where the ordering of the activations is the same as in γ, and *synchronizes* means that $p_{F_i}(x_i).F_i(x_i)$ and $\bar{p}_{F_k}\mathbf{m}'$ raise an internal communication.

In order to define when a vector of activations is correct for a process S with continuations $\vec{F} = (F_i)_{i\in 1\ldots n}$ and messages $\vec{\mathbf{m}} = (\mathbf{m}_i)_{i\in 1\ldots n}$ we consider, as a reference, the following vector $\mathscr{A}$ which is correct by construction:

$$\mathscr{A} = (F_i(\mathbf{m}_i) \mid i \in 1 \ldots n)$$

All the "correct" vectors of activations correspond to all the permutations of sub-vectors of $\mathscr{A}$.

Definition 6.5. Let $\mathscr{A}$ be defined as above. A vector Avs of activations is correct if and only if there exists an injective function $\theta : \{1,\ldots,|Avs|\} \mapsto \{1,\ldots,n\}$ such that $Avs_i = \mathscr{A}_{\theta(i)}$.

We define now the notion of *correct* delivering schema, which relates the activations of a computation γ to correct vectors.

Definition 6.6. Given a correct vector of activations Avs with injection θ, a *correct* delivering schema $\sigma : \{1,\ldots,|Avs|\} \mapsto \{1,\ldots,n\}$ of Avs and θ is defined as follows: $\sigma(k) = z$ if for some i we have $Avs_i = F_k(\mathbf{m}')$ and $\mathscr{A}_{\theta(i)} = F_z(\mathbf{m}_z)$.

By the fact that $Avs_i = \mathscr{A}_{\theta(i)}$ and that θ is injective, it is easy to see that a *correct* delivering schema σ of $Activation(\gamma)$ has the following relevant properties:

- σ is injective;
- for all $\bar{p}_{F_k}\mathbf{m}'$ synchronized in γ we have $\mathbf{m}_{\sigma(k)} = \mathbf{m}'$ and $F_k = F_{\sigma(k)}$.

Intuitively, this means that if a protocol activates a continuation through $\bar{p}_{F_k}\mathbf{m}'$, then σ is the function that returns the index $\sigma(k)$ of a correct delivering $F_{\sigma(k)}(\mathbf{m}_{\sigma(k)}) = F_k(\mathbf{m}')$, i.e., a delivering which can be executed by the secure specification. Moreover, each activation is mapped by σ into a different correct delivering.

The following lemma states that if a protocol executes a computation γ where only correct activations are performed, then also the secure specification can execute γ.

Lemma 6.7. *Let $S^{\vec{F}}(\vec{\mathbf{m}})$ be a protocol and T be a tester. Suppose $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$ through a computation γ such that $Activations(\gamma)$ is a correct vector of activations. Then, $S^{\vec{F}}_{\mathrm{spec}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$.*

As a direct consequence of the above lemma, we have that if $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$ and $S^{\vec{F}}_{\mathrm{spec}}(\vec{\mathbf{m}}) \parallel T \not\Downarrow \beta$, then, for all the successful computations γ of $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T$ (i.e., for all

the computations where $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T(\xrightarrow{\tau})^* \tilde{S}^{\vec{F}}(\vec{\mathbf{m}}) \parallel \tilde{T} \downarrow \beta$), we have that $Activations(\gamma)$ is not a correct vector of activations. This means that in every γ that leads T to success, we always have at least one message delivering which cannot be simulated by $S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}})$.

To prove the following result we need this additional well-formedness condition:

(WFC2) For every vector of messages $\vec{\mathbf{m}}$, $S(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$.

The condition above is an obvious requisite for S_{spec}: the process S and its specification S_{spec} behave in the same way when the public channels are protected through the restriction (i.e., when no attack is possible).

Lemma 6.8. *Let S be a protocol such that for every vector of messages $\vec{\mathbf{m}}$, $S(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$. If $S^{F'}$ guarantees TNDC-authentication then $S^{F'}$ guarantees weak spi-authentication.*

From the previous results the following can be easily proved.

Theorem 6.9. *Let S be a protocol satisfying the following well-formedness conditions*:

(WFC1) *For every vector of messages $\vec{\mathbf{m}}$, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k\in 1..n} F_k(\mathbf{m}_k)$.*
(WFC2) *For every vector of messages $\vec{\mathbf{m}}$, $S(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$.*

Then, S guarantees (strong) spi-authentication if and only if $S^{\vec{F}'}$ guarantees TNDC-authentication.

Proof. One direction follows from Corollary 6.3. By the other hand, if the process $S^{\vec{F}'}$ is authentic w.r.t. the *TNDC* based definition then by Lemma 6.8 we discover that $S^{\vec{F}'}$ is (weak) spi-authentic and finally by Proposition 3.9 we find out that S guarantees (strong) spi-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 a CryptoSPA process P is the product of the canonical continuations $\vec{F}'$ in order to obtain the final result (see Theorem 6.11).

Proposition 6.10. *Let P be a CryptoSPA process which can only execute a finite number of output actions and such that $ID(P) \subseteq \mathscr{D}(\phi_I)$. Then, for all $Q \in \mathscr{E}$, $P \approx_{\text{may}} Q$ iff $P \approx_{\text{trace}} Q$.*

Finally, the following result completes the comparison.

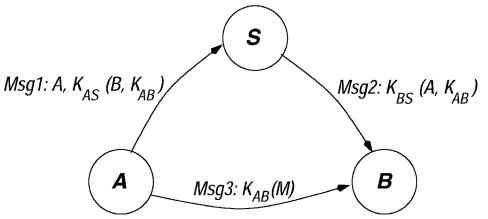

Fig. 6. Graphical description of the protocol.

Theorem 6.11. *Let S be a protocol which guarantees the well-formedness conditions* (WFC1) *and* (WFC2). *Then, S guarantees* (*strong*) *spi-authentication if and only if $S^{\vec{F}'}$ eiguarantees NDC-authentication.*

Proof. By Theorem 6.9, we need only to prove that $S^{\vec{F}'}(\vec{\mathbf{m}}) \in TNDC$ iff $S^{\vec{F}'}(\vec{\mathbf{m}}) \in NDC$. On the one hand, the fact that $S^{\vec{F}'}(\vec{\mathbf{m}}) \in NDC$ implies $S^{\vec{F}'}(\vec{\mathbf{m}}) \in TNDC$, follows from Proposition 2.6.

For the other direction, we have that

$$(S^{\vec{F}'}(\vec{\mathbf{m}}) \,\|\, X)\backslash C \approx_{\text{may}} S^{\vec{F}'}(\vec{\mathbf{m}})\backslash C$$

Now, since $S^{\vec{F}'}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k\in 1..n} F_k(\mathbf{m}_k)$ perform only a finite number of output actions and $ID(\prod_{k\in 1..n} F_k(\mathbf{m}_k)) \subseteq \mathscr{D}(\phi_I)$, by Proposition 6.10 we obtain that

$$(S^{\vec{F}'}(\vec{\mathbf{m}}) \,\|\, X)\backslash C \approx_{\text{trace}} S^{\vec{F}'}(\vec{\mathbf{m}})\backslash C$$

Hence, the thesis follows. □

6.1.1. An example: the Wide Mouthed Frog Protocol

In this section we show how to use NDC to analyze a simplified version (also studied in [2]) of the Wide Mouthed Frog Protocol [4].

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 (see also Fig. 6):

$$\begin{aligned}
&\text{Message 1} \quad A \to S : A, \{B, k_{AB}\}_{k_{AS}}\\
&\text{Message 2} \quad S \to B : \{A, k_{AB}\}_{k_{BS}}\\
&\text{Message 3} \quad A \to B : \{m_A\}_{k_{AB}}
\end{aligned}$$

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: [7]

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

where c_1, c_2 and c_3 are the three channels over which messages 1, 2 and 3 are communicated, respectively. Note that we have considered two instances of A and B in order to observe the replay attack. 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.

Let us see if the protocol guarantees the well-formedness conditions. First, it is easy to see that $P(\mathbf{m},\mathbf{m}')$ with no enemy, i.e., $P(\mathbf{m},\mathbf{m}')\backslash\{c_1,c_2,c_3\}$, is trace equivalent to $F(\mathbf{m}) \,\|\, F(\mathbf{m}')$. This represents the intended execution of the protocol. As a matter of fact the two sessions can be executed in any possible interleaving. It is also easy to prove that $P_{\text{spec}}(\mathbf{m},\mathbf{m}')\backslash\{c_1,c_2,c_3\} \approx_{\text{trace}} F(\mathbf{m}) \,\|\, F(\mathbf{m}')$. Thus, we finally have $P(\mathbf{m},\mathbf{m}')\backslash C \approx_{\text{trace}} P_{\text{spec}}(\mathbf{m},\mathbf{m}')\backslash C \approx_{\text{trace}} F(\mathbf{m}) \,\|\, F(\mathbf{m}')$.

Since $P(\mathbf{m},\mathbf{m}')$ is well formed, in order to check if it satisfies (strong) spi-authentication we can verify if $P^{\vec{F}'}(\mathbf{m},\mathbf{m}')$ guarantees NDC-authentication. Note that $P^{\vec{F}'}(\mathbf{m},\mathbf{m}')$ is obtained from $P(\mathbf{m},\mathbf{m}')$ 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 show that $P^{\vec{F}'}(\mathbf{m},\mathbf{m}')$ does not guarantee such a property. Consider the enemy $X \stackrel{\text{def}}{=} c_2(x).\overline{c_2}\,x.\overline{c_2}\,x.c_3(y).\overline{c_3}\,y.\overline{c_3}\,y$. It is easy to see that $(P \,\|\, X)\backslash C$ is able to execute the trace $\overline{out_F}\,\mathbf{m}.\overline{out_F}\,\mathbf{m}$ that is not a trace for process $\overline{out_F}\,\mathbf{m} \,\|\, \overline{out_F}\,\mathbf{m}'$. Hence, $(P^{\vec{F}'}(\mathbf{m},\mathbf{m}') \,\|\, X)\backslash C \not\approx_{\text{trace}} \overline{out_F}\,\mathbf{m} \,\|\, \overline{out_F}\,\mathbf{m}' \approx_{\text{trace}} P^{\vec{F}'}(\mathbf{m},\mathbf{m}')\backslash C$ and NDC-authentication is not satisfied.

The enemy X intercepts messages 2 and 3 and replays them, inducing B to commit twice on message $\mathbf{m}$ (as shown by trace $\overline{out_F}\,\mathbf{m}.\overline{out_F}\,\mathbf{m}$). This attack is quite critical in some situations. As an example, $\mathbf{m}$ could be a request of money transfer that would be executed twice. In order to avoid this attack, it is possible to modify the protocol (as done in [2]) by adding nonce-based challenge responses.

6.2. Agreement and spi-authentication

In this section, we compare the *Agreement* property and the spi-authentication one. We show that, in a particular (but reasonable) situation, *Agreement* implies spi-authentication. The intuition is that *Agreement* provides a form of authentication of the origin

[7] 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.

of a message, while the spi-authentication notion is more concerned with the integrity of the message itself.

First of all, we must choose a common setting for the comparison. We keep the same assumption we made for the previous comparison. Hence, we suppose that our processes are in the normal form (1) and satisfy the Assumptions 1 and 2, required by our previous analysis.

We are studying protocols which typically must deliver information (messages) among parties. In the *Agreement* property the roles of the sender (the initiator) and the receiver (the responder) are made explicit by using control actions. In the spi-authentication approach these roles are in a sense implicit. As a matter of fact, we can think of the processes with the continuations as the receivers, and of the other ones as the senders.

Typically, in the automated analysis of security protocols, fixed instances of sessions are considered, where specific messages are sent. We thus focus our comparison on this restricted framework and we consider a simple situation where a process is willing to independently receive n messages from several users. Thus, we assume to have n (not necessarily distinct) senders. We can describe this process in the following way:

$$S(x_1,\ldots,x_n) = \left(A_1(x_1) \parallel \cdots \parallel A_n(x_n) \parallel B^n \parallel \prod_n p_F(y).F(y)\right) \setminus \vec{p} \tag{4}$$

where B^n is the parallel composition of n copies of B in which the abstraction $F(y)$ is substituted by $\overline{p_F}(y)$.

For the analysis of the *Agreement* property we need to consider a slightly modified process, where control actions are inserted:

$$S'(x_1,\ldots,x_n) = \left(A'_1(x_1) \parallel \cdots \parallel A'_n(x_n) \parallel (B')^n \parallel \prod_n p_F(y).F(y)\right) \setminus \vec{p}$$

where for $i \in \{1,\ldots,n\}$ we have

$$A'_i(x_i) = \mathit{running\_ini}(A'_i, B', x_i).A(x_i)$$

and B' is obtained by suitably changing B so that each continuation $F(d)$, delivering message d, is preceded by the issuing of $\mathit{commit\_res}(B', A'_i, d)$ for some suitable $i \in \{1,\ldots,n\}$.

It is possible to prove that, in the situation we are considering, the *Agreement* property is strictly stronger than spi-authentication. Intuitively, this is due to the fact that spi-authentication records which messages are sent to the continuations, not their originators. In order to prove this result, we exploit the characterization of spi-authentication through non-interference described earlier.

Theorem 6.12. *Consider a process* $S(d_1,\ldots,d_n)$ *in the form* (4) *that satisfies the well-formedness conditions of* WFC1 *and* WFC2, *and consider the corresponding* $S'(d_1,\ldots,d_n)$. *If* $S'(d_1,\ldots,d_n)$ *satisfies* Agreement *then* $S(d_1,\ldots,d_n)$ *satisfies spi-authentication*.

We show now an interesting case of a process satisfying spi-authentication (once messages are fixed) but not *Agreement*. In particular, consider again the specification of the Wide Mouthed Frog protocol of Section 6.1.1 and fix the pair of messages to (m,m):

$$P' \stackrel{\text{def}}{=} (A(m,k_{AB}) \parallel A(m,k'_{AB}) \parallel B \parallel B \parallel S \parallel p_F(z).F(z) \parallel p_F(z).F(z)) \backslash p_F$$

Then, clearly spi-authentication is not able to detect the replay attack shown before, as the two messages are identical and the trace where m is delivered twice is just a legal trace. On the other hand, the *Agreement* property is not satisfied by this specification. In fact, the replay attack may be successfully completed even without the starting of the second run of the protocol, i.e. simply by emitting a single $running\_ini(A,B,m)$. Thus, producing a trace that is not possible for the specification $\alpha_{\text{Agree}}(S')$.

This gives an idea of the fact that spi-authentication, differently from *Agreement*, is not related to the actual originators of the messages. However, we should remark that when the quantification is performed over every vector of messages then the previous process satisfies neither NDC nor spi-authentication. Simply, we can consider a pair (m,m') where $m \neq m'$. This example shows that the quantification over all messages plays a central role in spi-authentication.

Example 6.13. In the following, we present two protocols which enjoy spi-authentication but not *Agreement*, thus proving that *Agreement* is strictly stronger than spi-authentication.

The first example is very simple. Alice sends to Bob a message x that Bob already knows. When Bob receives a message, he just compare it with x and proceeds only if they are equal:

$$S(x) = A(x) \parallel B(x)$$
$$A(x) = \bar{c}\,x$$
$$B(x) = c(y).[x = y]F(y)$$

Note that the only syntactic difference between S and S_{spec} is that $B(x) = c(y).[x = y]F(y)$ and $B_{\text{spec}}(x) = c(y).[x = y][x = y]F(x)$. Thus, for every $\mathbf{m}$ we have $B(\mathbf{m}) \approx_{\text{may}} B_{\text{spec}}(\mathbf{m})$ and so $S(\mathbf{m}) \approx_{\text{may}} S_{\text{spec}}(\mathbf{m})$, i.e., S satisfies spi-authentication.

Now, we refine $S(x)$ to perform the *Agreement* analysis as follows:

$$S'(x) = A'(x) \parallel B'(x)$$
$$A'(x) = running\_ini(A,B,x).\bar{c}\,x$$
$$B'(x) = c(y).[x = y]commit\_res(B,A,y).F(y)$$

If we consider an intruder which knows a message $\mathbf{m}$, e.g. $X = \bar{c}\,\mathbf{m}$, then $(S'(\mathbf{m}) \parallel X) \backslash C$ performs a trace where $commit\_res(B,A,\mathbf{m})$ is not preceded by $running\_ini(A,B,\mathbf{m})$. So, this process does not satisfy *Agreement*, but it satisfies spi-authentication.

Consider also a classical (insecure) authentication scheme based on password exchange: [8]

$$S(x) = A(x) \parallel B$$
$$A(x) = \bar{c}(x, pwd)$$
$$B = c(y).[y \vdash_{\text{fst}} y_1][y \vdash_{\text{snd}} y_2][y_1 = \mathbf{m}][y_2 = pwd]F(y_1)$$

Alice authenticates the message x to Bob by sending it with her password pwd. Bob knows Alice's password, and, after verifying that the received message is equal to $\mathbf{m}$, he checks whether the received password corresponds to pwd or not. (Bob could be a server whose unique client is Alice, and $\mathbf{m}$ would be the login name of Alice.)

Suppose the intruder has intercepted Alice's password in a previous session and so assume $pwd \in \mathcal{D}(\phi_I)$. As Bob always checks the received message against the expected one $\mathbf{m}$, it is impossible for an intruder to force him into accepting a faked message. With a reasoning similar to that in the previous example then we can show that S satisfies spi-authentication.

Now, we refine $S(x)$ to perform the *Agreement* analysis as follows:

$$\begin{aligned} S'(x) &= A'(x) \parallel B' \\ A'(x) &= running\_ini(A, B, x).\bar{c}(x, pwd) \\ B' &= c(y).[y \vdash_{\text{fst}} y_1][y \vdash_{\text{snd}} y_2][y_1 = \mathbf{m}][y_2 = pwd] \\ &\quad commit\_res(B, A, y_1).F(y_1) \end{aligned}$$

Consider an intruder who knows the message $\mathbf{m}$, e.g. $X = \bar{c}(\mathbf{m}, pwd)$ (recall that $pwd \in \mathcal{D}(\phi_I)$), then process $(S'(\mathbf{m}) \parallel X)\backslash C$ performs a trace where $commit\_res(B, A, \mathbf{m})$ is not preceded by $running\_ini(A, B, \mathbf{m})$. Again, this process does not satisfy *Agreement*, but it satisfies spi-authentication.

The two examples above, which discriminate between *Agreement* and spi-authentication, are somehow pathological as the receiver already knows the message to be received. Nevertheless, these examples clearly show a basic difference between the two properties. On the one hand, spi-authentication focuses only on verifying that no modified messages are accepted by the responder thus providing message authenticity/integrity. Here the intruder is sending the correct (expected) message, thus it does not represent an attack for spi-authentication. On the other hand, *Agreement* also verifies if the party on the other side of the network is the correct one, thus guaranteeing entity authentication. In Fig. 7 we summarize the relationships among the properties studied in this paper (where the subscript F' means that the property is checked for the process with the canonical continuation).

Example 6.14. In this example we show a direct application of the comparison results above. Consider again the specification of the Wide Mouthed Frog Protocol of Section 6.1.1. We could be interested to study if it satisfies the *Agreement* property. It is in the form (4). So if we consider the modified protocol $P'(m, m')$ where run and commit actions are added, we have that if $P'(m, m')$ satisfies *Agreement* then $P(m, m')$

[8] Note, however, that such a protocol does not enjoy the condition WFC1.

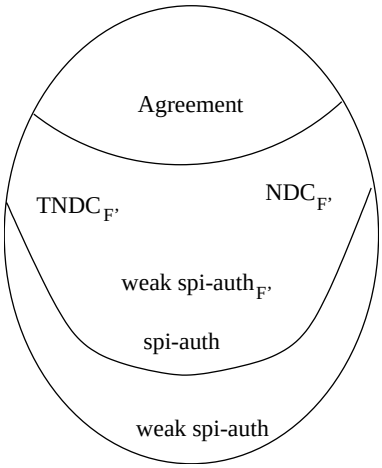

Fig. 7. Inclusion diagram among the authentication properties.

satisfies spi-authentication. Now, since we have shown that $P(m,m')$ does not satisfy spi-authentication then we can directly conclude that the corresponding $P'(m,m')$ does not satisfy *Agreement*. Following the proof of Theorem 6.12 we can also see that the attack on spi-authentication induces an attack on *Agreement* where the intruder is impersonating one of the two A processes.

7. Verification of authentication: the "most powerful enemy"

A common feature of the properties presented so far (including also spi-authentication) is that the verification has to be carried out against all the possible enemies/testers. One of the advantages of NDC is that it is possible to prove that a "most powerful enemy" exists. Thus, such properties may be verified just against such a particular hostile process.

In this section we briefly report the fundamental results about this "most powerful enemy" theory, without proving them. The interested reader may refer to [7,13,14].

A "most powerful enemy" can be defined by using a family of processes $Top^{C,\phi}_{\text{trace}}$ each representing the instance of the enemy with knowledge ϕ:

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

From the results in [13,14] we obtain the following:

Proposition 7.1. *Let S be a protocol. We have that*
(i) $S\in NDC$ iff $(S \parallel Top^{C,\phi_I}_{\text{trace}})\backslash C \approx_{\text{trace}} S\backslash C$;
(ii) *S satisfies Agreement iff* $(S \parallel Top^{\phi_I}_{\text{trace}})\backslash C \leqslant_{\text{trace}} \alpha_{\text{Agree}}(S)$.

8. Conclusion and future work

In order to compare various formalizations of security properties, we have defined in [13,12,14] a general scheme that allows us to capture a number of properties (e.g., authentication, secrecy, non-repudiation, and so on) as particular instances of the scheme itself. The results presented here are a first step towards our goal of formal comparison of security properties. Our aim is to obtain a complete classification in the style of [9,18], which could help in evaluating the relative merits of all such properties.

In this paper, we have compared under some reasonable assumptions three notions of authentication: spi-authentication, based on testing-equivalence, *Agreement*, based on the observation of correspondence actions, and NDC-authentication, based on non-interference. We have proved that the notion of spi-authentication may be equivalently expressed through the notion of NDC-authentication. Moreover, we have shown that the *Agreement* notion is at least as discriminating as the spi-authentication one. We have also identified some (very particular) cases in which spi-authentication is strictly weaker than *Agreement*. Roughly speaking, spi-authentication seems to be mostly concerned with the integrity of the data transmitted whereas *Agreement* has a stronger commitment on the authentication of the sender. We claim that for "relevant" protocols the two properties could collapse and we leave as future work the formal characterization of such protocols.

A main goal of our current research is to provide general, effective analysis 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 allows us, in principle, the reuse of automatic checking techniques for NDC in order to check spi-authentication over CryptoSPA protocols; indeed, as explained in [13,12,14] and briefly reported in Section 7, 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 $(S^{\vec{F}'}(\mathbf{m}_1,\ldots,\mathbf{m}_n) \parallel Top)\backslash C \approx_{\text{trace}} S^{\vec{F}'}(\mathbf{m}_1,\ldots,\mathbf{m}_n)\backslash C$ that can be verified using the CoSeC/CVS technology [6,7]. We are also investigating the problem of finding a finite set of vectors of messages, which could avoid the universal quantification over all the possible messages $\vec{\mathbf{m}}$. Moreover, it would be interesting to study general proof techniques for the analysis of unbounded processes and compare them with the ones proposed for authentication in the spi-calculus model (see, e.g., the proof of correctness of the Wide Mouthed Frog protocol in [1]).

Finally, 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 a lot the comparison, but we feel that at least some of the results presented here hold in this model as well.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments for the presentation of this paper.

Appendix Proof

Proposition 2.7. *Let $P, Q \in \mathcal{E}$ be two processes, $R \in \mathcal{E}^{\phi_I}$ a tester and L a set of channels. If $P \leqslant_{\text{may}} Q$ then*
(i) $P \parallel R \leqslant_{\text{may}} Q \parallel R$;
(ii) $P\backslash L \leqslant_{\text{may}} Q\backslash L$.

Proof. (i) We have to prove that for every barb β and for every process $T \in \mathcal{E}^{\phi_I}$, $(P \parallel R) \parallel T \Downarrow \beta$ implies that $(Q \parallel R) \parallel T \Downarrow \beta$. Now, if $(P \parallel R) \parallel T \Downarrow \beta$ then we have $(P \parallel R) \parallel T \Rightarrow (P' \parallel R') \parallel T'$ and also $(P' \parallel R') \parallel T' \downarrow \beta$. But due the operational semantics of CryptoSPA, $P \parallel (R \parallel T) \Rightarrow P' \parallel (R' \parallel T')$ and $P' \parallel (R' \parallel T') \downarrow \beta$. This means that P passes the test $(R \parallel T, \beta)$. Note that this is a correct test since $R \in \mathcal{E}^{\phi_I}$ and so also $R \parallel T \in \mathcal{E}^{\phi_I}$. Since $P \leqslant_{\text{may}} Q$, we have that also Q passes $(R \parallel T, \beta)$ and hence the thesis follows.

(ii) We have to prove that for every barb β and for every process $T \in \mathcal{E}^{\phi_I}$, $(P\backslash L) \parallel T \Downarrow \beta$ implies that $(Q\backslash L) \parallel T \Downarrow \beta$. If $\beta \in L$ then β is executed by T and we just consider a test T' where β is relabeled to $\beta' \notin L \cup sort(P) \cup sort(Q) \cup sort(T)$. Otherwise let β' be the same as β. Now we clearly have that $(P\backslash L) \parallel T' \Downarrow \beta'$. Moreover, since $(P\backslash L)$ cannot synchronize over channels in L then also $(P\backslash L) \parallel (T'\backslash L) \Downarrow \beta'$. Now, this clearly implies that also $P \parallel (T'\backslash L) \Downarrow \beta'$ and by hypothesis $Q \parallel (T'\backslash L) \Downarrow \beta'$. Since the test $(T'\backslash L)$ does not synchronize over channels in L we also have that $(Q\backslash L) \parallel (T'\backslash L) \Downarrow \beta'$ and clearly $(Q\backslash L) \parallel T' \Downarrow \beta'$. It is easy to see that also $(Q\backslash L) \parallel T \Downarrow \beta$. □

Proposition 2.11. $\precsim$, $\lessapprox$ $\subseteq \leqslant_{\text{may}}$.

Proof. As usual we have that $\precsim \subseteq \lessapprox$. Hence, we will prove that $\lessapprox \subseteq \leqslant_{\text{may}}$. Suppose that $P \lessapprox Q$, and that P passes a test (T, β), with $T \in \mathcal{E}^{\phi_I}$. This means that $P \parallel T \Rightarrow P' \parallel T'$ and $P' \parallel T' \downarrow \beta$. Since, $P \lessapprox Q$ we get that there exists $Q' \parallel T''$ such that $Q \parallel T \Rightarrow Q' \parallel T''$ and $P' \parallel T' \trianglelefteq Q' \parallel T''$. Thus, it follows that $Q' \parallel T'' \Downarrow \beta$ and finally the thesis. □

Proposition 2.15. *If $\mathcal{S}$ is a barbed simulation up to $\equiv$ then $\equiv \mathcal{S} \equiv$ is a barbed simulation.*

Proof. Let $P \equiv \mathcal{S} \equiv Q$, then there exist $P' \equiv P$ and $Q' \equiv Q$ such that $(P', Q') \in \mathcal{S}$. Now, if $P \downarrow \beta$, by Lemma 2.13 we have that $P' \downarrow \beta$ and, since $(P', Q') \in \mathcal{S}$, also $Q' \downarrow \beta$ and, by Lemma 2.13, also $Q \downarrow \beta$. Moreover, if $P \xrightarrow{\tau} \tilde{P}$ then also $P' \xrightarrow{\tau} \tilde{P}'$ with $\tilde{P} \equiv \tilde{P}'$. Since $(P', Q') \in \mathcal{S}$, then $Q' \xrightarrow{\tau} \tilde{Q}'$ with $(\tilde{P}', \tilde{Q}') \in \mathcal{S}$ and, by Lemma 2.13 $Q \xrightarrow{\tau} \tilde{Q}$ with $\tilde{Q}' \equiv \tilde{Q}$. As $\tilde{P} \equiv \tilde{P}'$, $(\tilde{P}', \tilde{Q}') \in \mathcal{S}$ and $\tilde{Q}' \equiv \tilde{Q}$ we get that $\tilde{P} \equiv \mathcal{S} \equiv \tilde{Q}$ and so the thesis. □

Lemma 3.4. *For all vector of messages $\vec{\mathbf{m}} = (\mathbf{m}_i)_{i \in 1 \ldots n}$, $S_{\text{spec}}(\vec{\mathbf{m}}) \precsim S(\vec{\mathbf{m}})$.*

Proof. The following is a barbed simulation up to $\equiv$:

$$\mathscr{S} = \left\{ \left(\left(Q \parallel \prod_{k\in I} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \setminus \vec{p} \parallel R \right), \right.$$
$$\left. \left(Q \parallel \prod_{k\in I} p_{F_k}(x).F_k(x) \right) \setminus \vec{p} \parallel R_1 \right) :$$
$$I \subseteq \{1,\ldots,n\},\ \vec{p} = \{p_{F_i}\}_{i\in 1\ldots n},\ \{Q,R,R_1\} \subseteq \mathscr{E},\ R \precsim R_1 \}$$

To prove this, consider $(A,B) \in \mathscr{S}$, then:

- if $A \downarrow \beta$ then we may have the following cases:
  - if $Q \downarrow \beta$ then also $B \downarrow \beta$;
  - if $R \downarrow \beta$ then also $B \downarrow \beta$, since $R \precsim R_1$;
- if $A \xrightarrow{\tau} A'$ then we may have the following cases:
  - if $Q \xrightarrow{\tau} Q'$ then also $B \xrightarrow{\tau} B'$ for some B' with $(A',B') \in \mathscr{S}$;
  - if $R \xrightarrow{\tau} R'$ then also $B \xrightarrow{\tau} B'$ for some B' with $(A',B') \in \mathscr{S}$ since $R \precsim R_1$;
  - if Q and $p_{F_j}(x).[x = \mathbf{m}_j]F_j(\mathbf{m}_j)$ synchronize, then it must be $Q \xrightarrow{\overline{p_{F_i}}\,\mathbf{m}'} Q'$ with $F_j = F_i$. We may have two cases depending on $\mathbf{m}'$:

    $\mathbf{m}' = \mathbf{m}_j$ then

$$A' \equiv \left(Q' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \setminus \vec{p} \parallel (F_j(\mathbf{m}_j) \parallel R)$$

    Then also $B \xrightarrow{\tau} B'$ and

$$B' \equiv \left(Q' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).F_k(x) \right) \setminus \vec{p} \parallel (F_j(\mathbf{m}') \parallel R_1)$$

    Since Q may synchronize with $p_{F_j}(x).F_j(x)$. Moreover, as $\mathbf{m}' = \mathbf{m}_j$ we have that $F_j(\mathbf{m}') \equiv F_j(\mathbf{m}_j)$. Thus, $A' \equiv \mathscr{S} \equiv B'$.

    $\mathbf{m}' \neq \mathbf{m}_j$ then

$$A' \equiv \left(Q' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \setminus \vec{p} \parallel R)$$

    Since Q may synchronize with $p_{F_j}(x).F_j(x)$, then also $B \xrightarrow{\tau} B'$ where

$$B' \equiv \left(Q' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).F_k(x) \right) \setminus \vec{p} \parallel (F_j(\mathbf{m}') \parallel R_1)$$

    Since $R \precsim F_j(\mathbf{m}') \parallel R_1$ we have $A' \equiv \mathscr{S} \equiv B'$.

Finally note that for all R, $(S_{\mathrm{spec}}(\vec{\mathbf{m}}) \parallel R, S(\vec{\mathbf{m}})) \parallel R) \in \mathscr{S}$. Thus, $S_{\mathrm{spec}}(\vec{\mathbf{m}}) \precsim S(\vec{\mathbf{m}})$. □

Proposition 3.9. *S guarantees spi-authentication iff $S^{\vec{F}'}$ guarantees weak spi-authentication, with $F_i'(x_i) = \overline{out_{F_i^*}}(x_i)$.*

Proof. One direction is obvious as $\vec{F}'$ is compatible with $\vec{F}^*$. Thus, if process S guarantees spi-authentication, then $S^{\vec{F}'}$ will certainly guarantee weak spi-authentication.

For the other direction, the proof is by contradiction. Assume that S does not guarantee spi-authentication; then we will arrive at the contradiction that also $S^{\vec{F}'}$ does not guarantee the weak spi-authentication property. We suppose that it is possible to find a vector of continuations $\vec{F}$ and a vector of messages $\vec{\mathbf{m}}$ such that $S^{\vec{F}}(\vec{\mathbf{m}}) \not\approx_{\text{may}} S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}})$. By Lemma 3.4, we have that $S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \leqslant_{\text{may}} S^{\vec{F}}(\vec{\mathbf{m}})$, so the assumption that $S^{\vec{F}}(\vec{\mathbf{m}}) \not\approx_{\text{may}} S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}})$ implies that $S^{\vec{F}}(\vec{\mathbf{m}}) \not\leqslant_{\text{may}} S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}})$. Thus, there exists an experiment (T, β) which is passed by $S^{\vec{F}}(\vec{\mathbf{m}})$ and is not passed by $S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}})$. Formally

$$S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta, \quad S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T \not\Downarrow \beta$$

Without loss of generality, we can assume that the channels of $\vec{F}'$ are neither in the sort of T nor in the one of $\vec{F}$. Indeed, in such a case, we could rename the channels in collision with the ones in $\vec{F}'$, obtaining another vector of continuations $\bar{F}$ which enjoys our requirements and such that $S^{\bar{F}}$ does not satisfy weak spi-authentication.

In principle, it could happen that F_j helped the attacker in such a way that cannot be simulated by F'_j. However, F_j can communicate nothing to the intruder that the intruder could not know. In fact, due to Assumption 2, we know that the processes $F_j(\mathbf{m}_j)$ have no secrets in collision with the protocol part. Hence the interaction of the protocol with F_j can be completely simulated by the intruder which may produce a copy of $F_j(\mathbf{m}_j)$.

Thus, consider the following tester T' as

$$\prod_{i \in 1 \ldots n} out_{F_i^*}(x_i).F_i(x_i) \parallel T$$

and let *Outs* be the set $\{out_{F_i^*}\}_{i \in 1 \ldots n}$.

Assume for a while that the following two inequalities hold:

(i) $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \lessapprox (S^{\vec{F}'}(\vec{\mathbf{m}}) \parallel T') \backslash Outs$,

(ii) $(S^{\vec{F}'}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T') \backslash Outs \lessapprox S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T$.

From (i) the following holds:

$$S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \lessapprox (S^{\vec{F}'}(\vec{\mathbf{m}}) \parallel T') \backslash Outs \lessapprox S^{\vec{F}'}(\vec{\mathbf{m}}) \parallel T'$$

Since $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$, with $\beta \notin Outs$, then also $S^{\vec{F}'}(\vec{\mathbf{m}}) \parallel T' \Downarrow \beta$. As, by hypothesis, $S^{\vec{F}'}$ satisfies weak spi-authentication, we derive $S^{\vec{F}'}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T' \Downarrow \beta$. Since $\beta \notin Outs$, also $(S^{\vec{F}'}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T') \backslash Outs \Downarrow \beta$. Now, by inequality (ii), we have also that $S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$, which leads to a contradiction (S would guarantee spi-authentication).

We now prove the two inequalities (i) and (ii).

For proving $S^{\vec{F}}(\vec{\mathbf{m}}) \parallel T \lessapprox (S^{\vec{F}'}(\vec{\mathbf{m}}) \parallel T') \backslash Outs$, let us show that the following is a barbed weak simulation up to $\equiv$:

$$\mathscr{S} = \left\{ \left(Q \parallel \prod_{k \in I} p_{F_k}(x).F_k(x) \right) \backslash \vec{p_I} \parallel T, \right.$$

$$\left(\left(Q_1 \parallel \prod_{k\in I} p_{F'_k}(x).F'_k(x)\right)\setminus \vec{p'_I} \parallel \prod_{i\in I} out_{F_i^*}(x).F_i(x)\right)\setminus Outs \parallel T):$$
$$I\subseteq\{1,\ldots,n\}, \vec{p_I}=\{p_{F_i}\}_{i\in I}, \vec{p'_I}=\{p_{F'_i}\}_{i\in I}, Q\in\mathscr{E}, Q_1=Q[f], T\in\mathscr{E}\Big\}$$

where f is the relabeling function such that $f(p_{F_i})=p_{F'_i}$.

Assume that $(A,B)\in\mathscr{S}$ then:

- if $A\downarrow\beta$ then we may have the following cases:
  - $Q\downarrow\beta$, in this case also $B\downarrow\beta$,
  - $T\downarrow\beta$, in this case also $B\downarrow\beta$.
- if $A\xrightarrow{\tau}A'$ then we may have the following cases:
  - $Q\xrightarrow{\tau}Q'$ then also $B\xrightarrow{\tau}B'$ for some B' and $(A',B')\in\mathscr{S}$,
  - $T\xrightarrow{\tau}T'$ then also $B\xrightarrow{\tau}B'$ for some B' and $(A',B')\in\mathscr{S}$,
  - if there is a synchronization between $Q\xrightarrow{\overline{p_{F_i}}(\mathbf{m}')}Q'$ and the prefixed continuation $p_{F_j}(x).F_j(x)$ then let $J=I\setminus\{j\}$ and

$$A'\equiv\left(Q'\parallel\prod_{k\in J}p_{F_k}(x).F_k(x)\right)\setminus\vec{p_J}\parallel F_j(\mathbf{m}')\parallel T$$

We also have that $B\xrightarrow{\tau}B'$ where

$$B'\equiv\left(Q'_1\parallel\prod_{k\in J}p_{F'_k}(x).F'_k(x)\right)\setminus\vec{p'_J}\parallel\overline{out_{F_j^*}}\,\mathbf{m}'\parallel\prod_{i\in I}out_{F_i^*}(x).F_i(x))\setminus Outs\parallel T$$

where $Q'_1=Q'[f]$ since $F'_j=\overline{out_{F_j^*}}$. Thus, we have that $B'\xrightarrow{\tau}B''$ by means of a synchronization on the channel $out_{F_j^*}$ where

$$B''\equiv\left(Q'_1\parallel\prod_{k\in J}p_{F'_k}(x).F'_k(x)\right)\setminus\vec{p'_J}\parallel\prod_{i\in J}out_{F_i^*}(x).F_i(x))\setminus Outs\parallel F_j(\mathbf{m}')\parallel T$$

and $(A',B'')\in\mathscr{S}$.

For proving $(S^{\vec{F}'}_{\mathrm{spec}}(\vec{\mathbf{m}})\parallel T')\setminus Outs\lessapprox S^{\vec{F}}_{\mathrm{spec}}(\vec{\mathbf{m}})\parallel T$ let us show that the following is a barbed weak simulation (up to $\equiv$):

$$\mathscr{S}=\Bigg\{\left(\left(Q\parallel\prod_{k\in I_1}p_{F'_k}(x).[x=\mathbf{m}_k]F'_k(\mathbf{m}_k)\right)\setminus\vec{p'_{I_1}}\parallel\prod_{k'\in I_2}\overline{out_{F_{k'}^*}}(\mathbf{m}_{k'})\parallel\prod_{i\in I}out_{F_i^*}(x).F_i(x)\right)\setminus Outs\parallel T,$$
$$\left(Q_1\parallel\prod_{k\in I_2}\overline{p_{F_k}}\,\mathbf{m}_k\parallel\prod_{k\in I}p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k)\right)\setminus\vec{p_I}\parallel T:$$

$$I_1 \cup I_2 \subseteq I \subseteq \{1,\ldots,n\},\ I_1 \cap I_2 = \emptyset,\ \vec{p_I} = \{p_{F_i}\}_{i\in I},\ \vec{p'_{I_1}} = \{p_{F'_i}\}_{i\in I_1},$$
$$Q \in \mathscr{E},\ T \in \mathscr{E},\ Q_1 = Q[f']\Big\}$$

where f' is the relabeling function such that $f'(p_{F'_i}) = p_{F_i}$. Assume that $(A,B) \in \mathscr{S}$ then:

- if $A \downarrow \beta$ then we may have the following cases:
  - $Q \downarrow \beta$, in this case also $B \downarrow \beta$,
  - $T \downarrow \beta$, in this case also $B \downarrow \beta$.
- if $A \xrightarrow{\tau} A'$ then we may have the following cases:
  - $Q \xrightarrow{\tau} Q'$ then also $B \xrightarrow{\tau} B'$ for some B' and $(A',B') \in \mathscr{S}$,
  - $T \xrightarrow{\tau} T'$ then also $B \xrightarrow{\tau} B'$ for some B' and $(A',B') \in \mathscr{S}$,
  - if there is a synchronization between a term $\overline{out_{F_k^*}}(\mathbf{m}_k)$ with $k \in I_2$ and a prefixed continuation $out_{F_l^*}(x).F_l(x)$ with $l \in I$, then let $K = I\backslash\{k\}$:

$$A' \equiv \Bigg(Q \parallel \prod_{k\in I_1} p_{F'_k}(x).[x = \mathbf{m}_k]F'_k(\mathbf{m}_k)\backslash \vec{p'_{I_1}} \parallel \prod_{k'\in I_2\backslash\{k\}} out_{F_{k'}^*}(\mathbf{m}_{k'})$$
$$\parallel \prod_{i\in K} out_{F_i^*}(x).F_i(x)\Bigg)\backslash Outs \parallel F_l(\mathbf{m}_k) \parallel T$$

We also have $B \xrightarrow{\tau} B'$ by means of a synchronization on p_{F_k} with

$$B' \equiv \Bigg(Q_1 \parallel \prod_{i\in I_2\backslash\{k\}} \overline{p_{F_i}}\,\mathbf{m}_i \parallel \prod_{h\in K} p_{F_h}(x).[x = \mathbf{m}_h]F_h(\mathbf{m}_h)\Bigg)\backslash \vec{p_K} \parallel$$
$$\parallel F_k(\mathbf{m}_k) \parallel T$$

since $F_l(\mathbf{m}_k) = F_k(\mathbf{m}_k)$ then we have that $(A',B') \in \mathscr{S}$,

  - if there is a synchronization between $Q \xrightarrow{\overline{p_{F'_j}}(\mathbf{m}')} Q'$ and the prefixed continuation $p_{F'_j}(x).[x = \mathbf{m}_j]F'_j(\mathbf{m}_j)$ then let $J_1 = I_1\backslash\{j\}$, $J = I\backslash\{j\}$ and we may have two cases depending on $\mathbf{m}'$:

    $\mathbf{m}' = \mathbf{m}_j$. Then we have

$$A' \equiv \Bigg(Q' \parallel \prod_{k\in J_1} p_{F'_k}(x).[x = \mathbf{m}_k]F'_k(\mathbf{m}_k)\backslash \vec{p'_{J_1}} \parallel$$
$$\parallel \prod_{k'\in I_2\cup\{j\}} \overline{out_{F_{k'}^*}}(\mathbf{m}_{k'}) \parallel \prod_{i\in I} out_{F_i^*}(x).F_i(x)\Bigg)\backslash Outs \parallel T$$

We have also that $B \xrightarrow{\tau} B'$ where

$$B' \equiv \Bigg(Q'_1 \parallel \prod_{k\in I_2\cup\{j\}} \overline{p_{F_k}}\,\mathbf{m}_k \parallel \prod_{k\in J} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k)\Bigg)\backslash \vec{p_J}$$
$$\parallel T$$

since $Q \equiv Q' \parallel \overline{p_{F_j'}}(\mathbf{m}_j)$ and so $Q_1 \equiv Q[f'] \parallel \overline{p_{F_j}}(\mathbf{m}_j)$ (recall $p_{F_j'} = p_{F_i'}$ and $f'(p_{F_j'}) = p_{F_j}$) and finally $(A', B') \in \mathscr{S}$.

$\mathbf{m}' \neq \mathbf{m}_j$. Then we have

$$A' \equiv \left(Q' \parallel \prod_{k \in J_1} p_{F_k'}(x).[x = \mathbf{m}_k]F_k'(\mathbf{m}_k) \backslash \vec{p_{J_1}'} \parallel \right.$$

$$\left. \parallel \prod_{k' \in I_2} \overline{out_{F_{k'}^*}}(\mathbf{m}_{k'}) \parallel \prod_{i \in I} out_{F_i^*}(x).F_i(x) \right) \backslash Outs \parallel T$$

and also $B \xrightarrow{\tau} B'$ where

$$B' \equiv \left(Q_1' \parallel \prod_{k \in I_2} \overline{p_{F_k}}\,\mathbf{m}_k \parallel \prod_{k \in J} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \backslash \vec{p_J} \parallel T$$

and $(A', B') \in \mathscr{S}$. □

Lemma 6.2. *Assume that for every vector of messages* $\vec{\mathbf{m}}$, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} \prod_{k \in 1..n} F_k(\mathbf{m}_k)$. *Then,* S_{spec} *guarantees TNDC-authentication.*

Proof. We have to prove that $(S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$. One direction is trivial, as $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \leqslant_{\text{trace}} (S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C$ and so, by Proposition 2.6, $S_{\text{spec}}(\vec{\mathbf{m}})\backslash C \leqslant_{\text{may}} (S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C$. Let us prove that

$$(S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C \leqslant_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C.$$

By the hypothesis we have that $\prod_{k \in 1...n} F_k(\mathbf{m}_k) \leqslant_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$. Thus, if we prove that $(S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C \leqslant_{\text{may}} \prod_{k \in 1...n} F_k(\mathbf{m}_k)$, then we are done. We do this by showing that $(S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C \lessapprox \prod_{k \in 1...n} F_k(\mathbf{m}_k)$ from which the result follows by Proposition 2.11.

First of all, as $S_{\text{spec}}(\vec{\mathbf{m}})$ is in form (2), we note that

$$(S_{\text{spec}}(\vec{\mathbf{m}}) \parallel X)\backslash C \equiv$$
$$\left(Q \parallel X \parallel \prod_{k \in 1...n} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \backslash \vec{p} \backslash C \equiv$$
$$\left((Q \mid X)\backslash C \parallel \prod_{k \in 1...n} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \backslash \vec{p}$$

for some $Q \in \mathscr{E}_{C \cup \vec{p}}$ since, by definition, the channels of the continuations are different from the ones in the protocol (see Note 1), i.e. they are different from $C \cup \vec{p}$.

Let us consider the following relation:

$$\mathscr{S} = \left\{ \left(Q'\backslash C \parallel \prod_{k \in I} p_{F_k}(x).[x = \mathbf{m}_k]F_k(\mathbf{m}_k) \right) \backslash \vec{p_I} \parallel R, \prod_{k \in I} F_k(\mathbf{m}_k) \parallel R_1 \right):$$
$$\left. I \subseteq \{1, \ldots, n\},\ \vec{p_I} = \{p_{F_i}\}_{i \in I},\ Q' \in \mathscr{E}_{C \cup \vec{p_I}},\ \{R, R_1\} \subseteq \mathscr{E},\ R \lesssim R_1 \right\}$$

We prove that $\mathscr{S}$ is a *weak barbed simulation* up to $\equiv$. Suppose that $(A,B)\in\mathscr{S}$ then:

- if $A\downarrow\beta$ then it must be $R\downarrow\beta$ and so $B\downarrow\beta$ since $R\precsim R_1$,
- if $A\xrightarrow{\tau}A'$, because of $R\xrightarrow{\tau}R'$ then also $B\xrightarrow{\tau}B'$ since $R\precsim R_1$ and $A'\mathscr{S}B'$,
- if $A\xrightarrow{\tau}A'$ because of $Q'\xrightarrow{\tau}Q''$ then $A'\mathscr{S}B$,
- if $A\xrightarrow{\tau}A'$ because of a synchronization between Q' and one of the factors $p_{F_i}(x).[x=\mathbf{m}_i]F_i(\mathbf{m}_i)$ then we may have two cases:
  - $Q'\xrightarrow{\overline{p_{F_j}}\,\mathbf{m}'}Q''$ and $\mathbf{m}'=\mathbf{m}_i$. Let $J=I\backslash\{i\}$, then:

$$\begin{aligned} A' &\equiv \\ \left(Q''\backslash C \parallel \prod_{k\in J} p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k)\right)\backslash\vec{p_J} \parallel (F_i(\mathbf{m}_i)\parallel R) &\mathscr{S} \\ \prod_{k\in J} F_k(\mathbf{m}_k)\parallel(F_i(\mathbf{m}_i)\parallel R_1) &\equiv \\ \prod_{k\in I} F_k(\mathbf{m}_k)\parallel R_1 &\equiv \\ B & \end{aligned}$$

  - $Q'\xrightarrow{\overline{p_{F_j}}\,\mathbf{m}'}Q''$ and $\mathbf{m}'\neq\mathbf{m}_i$. Then we have

$$\begin{aligned} A' &\equiv \\ \left(Q''\backslash C \parallel \prod_{k\in J} p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k)\right)\backslash\vec{p_J} \parallel R &\mathscr{S} \\ \prod_{k\in J} F_k(\mathbf{m}_k)\parallel(F_i(\mathbf{m}')\parallel R_1) &\equiv \\ B & \end{aligned}$$

    since $R\precsim F_i(\mathbf{m}')\parallel R_1$.

So $\equiv\mathscr{S}\equiv$ is a *weak barbed simulation* and since we have that

$$(S_{\mathrm{spec}}(\vec{\mathbf{m}})\parallel X)\backslash C\equiv$$

$$\left((Q\parallel X)\backslash C\parallel\prod_{k\in 1\ldots n} p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k)\right)\backslash\vec{p}\ \mathscr{S}\prod_{k\in 1\ldots n}F_k(\mathbf{m}_k)$$

then $(S_{\mathrm{spec}}(\vec{\mathbf{m}})\parallel X)\backslash C\lessapprox\prod_{k\in 1\ldots n}F_k(\mathbf{m}_k)$. □

Lemma 6.7. *Let $S^{\vec{F}}(\vec{\mathbf{m}})$ be a protocol and T be a tester. Suppose $S^{\vec{F}}(\vec{\mathbf{m}})\parallel T\Downarrow\beta$ through a computation γ such that Activations(γ) is a correct vector of activations. Then, $S^{\vec{F}}_{\mathrm{spec}}(\vec{\mathbf{m}})\parallel T\Downarrow\beta$.*

Proof. By induction on the number n of activations in γ.

- $n=0$: This case is simple. Due to the form of processes and specifications, two processes $S^{\vec{F}}(\vec{\mathbf{m}})$ and $S^{\vec{F}}_{\mathrm{spec}}(\vec{\mathbf{m}})$ cannot exhibit a different behavior without performing some synchronizations on the channels in $\vec{p}$.
- $n+1$: Let p_{F_i} be the channel where the two processes perform the synchronization of the first activation in γ and let σ be a correct delivering scheme for $Activations(\gamma)$. Then

$$S^{\vec{F}}(\vec{\mathbf{m}})\parallel T\ \Rightarrow\ S'^{\vec{F}}(\vec{\mathbf{m}})\parallel T'$$

and

$$S'^{\vec{F}}(\vec{\mathbf{m}}) \equiv \left(\overline{p_{F_i}}(\mathbf{m}_{\sigma(i)}) \parallel p_{F_j}(x).F_j(x_j) \parallel P' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).F_k(x) \right) \setminus \vec{p}.$$

Thus, $S'^{\vec{F}}(\vec{\mathbf{m}}) \parallel T' \xrightarrow{\tau} S'' \parallel F_j(\mathbf{m}_{\sigma(i)}) \parallel T'$ where

$$S'' \equiv \left(P' \parallel \prod_{k\in I\setminus\{j\}} p_{F_k}(x).F_k(x) \right) \setminus \vec{p}.$$

Then, we can see that also

$$S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T \;\Rightarrow\; S'^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T'$$

where

$$S'^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \equiv \left(\overline{p_{F_i}}(\mathbf{m}_{\sigma(i)}) \parallel p_{F_{\sigma(i)}}(x).[x=\mathbf{m}_{\sigma(i)}]F_{\sigma(i)}(\mathbf{m}_{\sigma(i)}) \parallel P' \parallel \prod_{k\in I\setminus\{\sigma(i)\}} p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k) \right) \setminus \vec{p}.$$

Thus, also $S'^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T' \xrightarrow{\tau} S''_1 \parallel F_{\sigma(i)}(\mathbf{m}_{\sigma(i)}) \parallel T'$ where

$$S''_1 \equiv \left(P' \parallel \prod_{k\in I\setminus\{\sigma(i)\}} p_{F_k}(x).[x=\mathbf{m}_k]F_k(\mathbf{m}_k) \right) \setminus \vec{p}$$

Since σ is a correct delivering schema, we have that $F_{\sigma(i)} = F_i$, but we also know that $F_i = F_j$ thus it follows $F_{\sigma(i)} = F_j$. Let R' be $F_j(\mathbf{m}_{\sigma(i)}) = F_{\sigma(i)}(\mathbf{m}_{\sigma(i)})$.

Now, consider the process $S'''^{\vec{F}'}(\vec{\mathbf{m}}')$ obtained by considering P' as execution part and where $\vec{F}'$ is obtained from $\vec{F}$ by substituting $F_{\sigma(i)}$ with F_i by deleting the ith continuation; $\vec{\mathbf{m}}'$ is obtained from $\vec{\mathbf{m}}$ by substituting $\mathbf{m}_{\sigma(i)}$ with $\mathbf{m}_i$ and by deleting the ith-component of the resulting vector.

Then we can note that both

$$S'''^{\vec{F}'}(\vec{\mathbf{m}}') \equiv S''$$

and

$$S_{\text{spec}}{}'''^{\vec{F}'}(\vec{\mathbf{m}}') \equiv S''_1$$

in particular the first equivalence holds since $p_{F_j}(x).F_j(x)$ is equal to $p_{F_i}(x).F_i(x)$.

Then, it is easy to see that $S'''^{\vec{F}'}(\vec{\mathbf{m}}') \parallel F_j(\mathbf{m}_{\sigma(i)}) \parallel T' \Downarrow \beta$ with a successful computation γ' whose delivering schema σ' is correct and σ' is as σ restricted on $I\setminus\{i\}$ except for $\sigma'(l) = \sigma(i)$, where $\sigma(l) = i$ and $i \neq l$. Thus, we can apply the inductive hypothesis and we obtain that also $S_{\text{spec}}{}'''^{\vec{F}'}(\vec{\mathbf{m}}') \parallel F_{\sigma(i)}(\mathbf{m}_{\sigma(i)}) \parallel T' \Downarrow \beta$ by proving that $S^{\vec{F}}_{\text{spec}}(\vec{\mathbf{m}}) \parallel T \Downarrow \beta$. □

Lemma 6.8. *Let S be a protocol such that for every vector of messages $\vec{\mathbf{m}}$, $S(\vec{\mathbf{m}})\backslash C \approx_{\text{may}} S_{\text{spec}}(\vec{\mathbf{m}})\backslash C$. If $S^{F'}$ guarantees TNDC-authentication then $S^{F'}$ guarantees weak spi-authentication.*

Proof. By contradiction, we assume $S^{F'}$ does not guarantee weak spi-authentication. By Lemma 3.4, there exists an experiment (T,β) which is passed by $S^{\vec{F}'}(\vec{\mathbf{m}})$ and is not passed by $S^{\vec{F}'}_{\text{spec}}(\vec{\mathbf{m}})$.

By Lemma 6.7, it must be that at a certain point of the computation with the attacker T a configuration of the process S is reached such that one of the components sends a non-*correct* $\mathbf{m}'$ to a continuation F_i (according to Definitions 6.5 and 6.6).

Our aim is to split the tester T in an attacker X and an observer T' such that the attacker communicates only on the channels C and the observer only on the channels of the continuations. Because of the restriction on the sort of X we cannot do as in Proposition 3.9 to consider a different attacker who plays the role of some continuations F_i. This role will be played by our observer T'.

Now, consider the shortest computation between the intruder T and the process S which leads the processes to send an incorrect value to a continuation $F_{i_{l+1}}$. During this computation it is possible that the intruder T has committed with some continuation F'_j, i.e. it has received on some channel $out_{F^*_j}$. Let $B=(F'_{i_1}(\mathbf{m}'_{i_1}),\ldots,F'_{i_l}(\mathbf{m}'_{i_l}))$ be the ordered vector of channel/messages that the intruder has synchronized with. By our assumption on the fact that we consider the shortest computation which leads to a similar configuration we know that the intruder in these receptions has received only *correct* values.

As an attacker we consider a process X which is obtained from the process T where the receptions on the channels $out_{F^*_{i_j}}$ are removed, by replacing the relative input variables with the correct message $\mathbf{m}_{i_j}$.

Hence, we can write a process T' which receives on the barbs in B, then receives a value on the channel $out_{F^*_{i_{l+1}}}$ and then it tests whether the received value is different from the possible *correct* ones (we use an if–then–else operator) and if so it commits on a special channel, say ω.

$$
\begin{aligned}
T' = {} & (out_{F^*_{i_1}}(y).[y=\mathbf{m}'_{i_1}]\overline{go_{i_2}} \parallel \\
& \prod_{k\in\{2,\ldots,l\}} go_{i_k}.out_{F^*_{i_k}}(y).[y=\mathbf{m}'_{i_k}]\overline{go_{i_{k+1}}} \parallel \\
& go_{i_{l+1}}.out_{F^*_{i_{l+1}}}(y).[y=\mathbf{m}_{z_1}]\underline{0};[y=\mathbf{m}_{z_2}]\underline{0};\ldots[y=\mathbf{m}_{z_r}]\underline{0};\omega)\backslash\vec{go}
\end{aligned}
$$

where $\mathbf{m}_{z_1},\ldots,\mathbf{m}_{z_r}$ are the possible correct messages for $F_{i_{l+1}}$ after γ and $\vec{go}=\{go_{i_k}\}_{k\in\{1,\ldots,l+1\}}$ are synchronization channels. The restricted specification $S^{\vec{F}'}_{\text{spec}}(\vec{\mathbf{m}})\backslash C$ which is equivalent to $S^{\vec{F}'}(\vec{\mathbf{m}})\backslash C$ (by hypothesis) can never produce an incorrect message. Thus we have $(S^{\vec{F}'}(\vec{\mathbf{m}})\parallel X)\backslash C \parallel T' \Downarrow \omega$ while $S^{\vec{F}'}(\vec{\mathbf{m}})\backslash C \parallel T' \not\Downarrow \omega$. This shows the contradiction $S^{\vec{F}'}\notin TNDC$. □

Proposition 6.10. *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 $Q\in\mathcal{E}$, $P\approx_{\text{may}} Q$ iff $P\approx_{\text{trace}} Q$.*

Proof. By Proposition 2.6 we have that if $P \approx_{\text{trace}} Q$ then $P \approx_{\text{may}} Q$.

We now prove that if $P \leqslant_{\text{may}} Q$ then $P \leqslant_{\text{trace}} Q$. Consider a trace γ of P. Since $ID(P) \subseteq \mathscr{D}(\phi_I)$, then γ may contain only output messages that are in $\mathscr{D}(\phi_I)$. Hence, we can construct a test $T \in \mathscr{E}^{\phi_I}$ that executes a barb β iff the tested process can execute γ. It is sufficient that the test checks that every received message is the correct one through the matching operator. We have that $P \parallel T \Downarrow \beta$, and by hypothesis, also $Q \parallel T \Downarrow \beta$. This implies that, by construction of T, γ must be also a trace for Q. Indeed, as we already stated, T is such that every message of Q is checked against the messages contained in γ. T executes β only if every action of γ is executed.

We still have to prove that if $Q \leqslant_{\text{may}} P$ then $Q \leqslant_{\text{trace}} P$.

First, we show that Q can only provide a sequence of output messages in $\mathscr{D}(\phi_I)$. Try to suppose that Q performs an output of a secret message after, say n, outputs of public messages. In this case, we can built a process T which receives $n+1$ values. T composed with Q receives this value. If we compose this tester with P we must obtain that P may send a finite set of values $m_1, \ldots m_l$ that are not secret, after a sequence of n outputs. The situation where P is not able to send a sequence of $n+1$ messages should directly contradict assumption $Q \leqslant_{\text{may}} P$. Hence, we can simply modify T in such a way that it tests whether the last message received is different from the messages $m_1, \ldots m_l$, if so it commits on a special channel ω. Now, we get an absurd, since Q passes (T, ω) while P cannot do it.

Next, it is easy to see that Q cannot execute input actions. If it could do so, for example after a sequence of output of public messages, then we could consider a test that sends an output to Q and then executes on a special barb ω. This test cannot succeed with P, but as $Q \leqslant_{\text{may}} P$, we obtain a contradiction.

We have proved that Q may only perform outputs of public messages. Now, to show that $Q \leqslant_{\text{may}} P$ implies $Q \leqslant_{\text{trace}} P$ we can proceed as at the beginning of the proof. Hence, the thesis follows. □

Theorem 6.12. *Consider a process $S(d_1, \ldots, d_n)$ in the form (4) that satisfies the well-formedness conditions of* (WFC1) *and* (WFC2), *and consider the corresponding $S'(d_1, \ldots, d_n)$. If $S'(d_1, \ldots, d_n)$ satisfies* Agreement *then $S(d_1, \ldots, d_n)$ satisfies spi-authentication.*

Proof. Fix a vector of messages $(d_1, \ldots, d_n)$. We need to show that

$$\forall X \ (S'(d_1, \ldots, d_n) \parallel X) \backslash C \leqslant_{\text{trace}} \alpha_{\text{Agree}}(S'(d_1, \ldots, d_n)) \quad \text{implies}$$
$$S^{F'}(d_1, \ldots, d_n) \in NDC$$

as, by Lemma 6.8 and Proposition 6.10, $S^{F'}(d_1, \ldots, d_n) \in NDC$ implies spi-authentication.

By contradiction, suppose that $S^{F'}(d_1, \ldots d_n) \notin NDC$. Then it must be that for some intruder X the compound process $(S^{F'}(d_1, \ldots d_n) \parallel X) \backslash C$ is not equivalent to $S^{F'}(d_1, \ldots d_n) \backslash C$. Now note that, by the well-formedness condition (WFC1) we made on the process, we must have $S^{F'}(d_1, \ldots d_n) \backslash C \approx_{\text{trace}} \prod_{i \in 1 \ldots n} \overline{out_F}(d_i)$. Thus, it must be the case that $(S^{F'}(d_1, \ldots d_n) \parallel X) \backslash C$ produces an *incorrect* trace, i.e., which is not a trace

for $\prod_n \overline{out_F}(d_n)$. This happens if the ordered sequence of messages emitted is not a prefix of some permutation of the sequence $d_1, \ldots, d_n$.

But, by construction of S', we have that also $(S'(d_1, \ldots . d_n) \| X) \backslash C$ produces a sequence of action *commit_res* whose ordered sequence of messages is not correct. In fact, each activation of a continuation $F(d)$ is preceded by the issuing of the action $commit\_res(B, A_i, d)$ for some A_i. Now, the same trace is produced by the process $(S'(d_1, \ldots . d_n) \| X) \backslash C$. But, since this process satisfies *Agreement*, this means that in the incorrect trace there is also an incorrect sequence of messages delivered on the channels *running_ini*. This is impossible since it is easy to see that, by construction, $S'(d_1, \ldots . d_n)$ produces only correct sequences of outputs in this channel. □

References

[1] M. Abadi, A.D. Gordon, Reasoning about cryptographic protocols in the spi calculus, in: Proc. CONCUR'97, Lecture Notes in Computer Science, Vol. 1243, 1997, pp. 59–73.
[2] M. Abadi, A.D. Gordon, A calculus for cryptographic protocols : the spi calculus, Inform. and Comput. 148 (1) (1999) 1–70.
[3] C. Bodei, P. Degano, R. Focardi, C. Priami, Authentication via localized names, in: Proc. CSFW'99, IEEE Press, New York, 1999, pp. 98–110.
[4] M. Burrows, M. Abadi, R. Needham, A logic of authentication, Proc. Roy. Soc. London 426 (1989) 233–271.
[5] R. De Nicola, M. Hennessy, Testing equivalences for processes, Theoret. Comput. Sci. 34 (1984) 83–133.
[6] A. Durante, R. Focardi, R. Gorrieri, CVS: A compiler for the analysis of cryptographic protocols, in: Proc. CSFW'99, IEEE Press, New York, 1999, pp. 203–212.
[7] A. Durante, R. Focardi, R. Gorrieri, A compiler for analysing cryptographic protocols using non-interference, ACM Trans. Software Eng. Methodol. (TOSEM) 9 (4) (2000) 488–528.
[8] R. Focardi, A. Ghelli, R. Gorrieri, Using non interference for the analysis of security protocols, in: Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997.
[9] R. Focardi, R. Gorrieri, A classification of security properties for process algebras, J. Comput. Security 3 (1) (1994/1995) 5–33.
[10] R. Focardi, R. Gorrieri, The compositional security checker : a tool for the verification of information flow security properties, IEEE Trans. Software Eng. 23 (9) (1997) 550–571.
[11] R. Focardi, R. Gorrieri, F. Martinelli, Message authentication through non-interference, in: Proc. Eighth Internat. Conf. in Algebraic Methodology and Software Technology (AMAST 2000), Lecture Notes in Computer Science, Vol. 1816, Springer, Berlin, 2000, pp. 258–272.
[12] R. Focardi, R. Gorrieri, F. Martinelli, Non interference for the analysis of cryptographic protocols, in: U. Montanari (Ed.), Proc. 27th Internat. Coll. on Automata, Languages and Programming (ICALP'00), Lecture Notes in Computer Science, Vol. 1853, July 2000, pp. 354–372.
[13] R. Focardi, R. Gorrieri, F. Martinelli, A classification of security properties for systems and networks, forthcoming.
[14] R. Focardi, F. Martinelli, A uniform approach for the definition of security properties, in: Proc. World Congr. on Formal Methods (FM'99), Lecture Notes in Computer Science, Vol. 1708, Springer, Berlin, 1999, pp. 794–813.
[15] J.A. Goguen, J. Meseguer, Security policy and security models, in: Proc. 1982 Symp. on Security and Privacy, IEEE Press, New York, 1982, pp. 11–20.
[16] D. Gollman, What do we mean by entity authentication? in: Proc. Symp. in Research in Security and Privacy, IEEE Press, New York, 1996, pp. 46–54.
[17] G. Lowe, Breaking and fixing the Needham–Schroeder public-key protocol using FDR, in: Proc. TACAS'96, Lecture Notes in Computer Science, Vol. 1055, 1996, pp. 146–166.

[18] G. Lowe, A hierarchy of authentication specification, in: Proc. 10th Computer Security Foundation Workshop, IEEE Press, New York, 1997, pp. 31–43.
[19] W. Marrero, E. Clarke, S. Jha, A model checker for authentication protocols, in: Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, Rutgers University, September 1997.
[20] F. Martinelli, Languages for description and analysis of authentication protocols, in: Proc. ICTCS'98, World Scientific, Singapore, 1998, pp. 304–315.
[21] F. Martinelli, Partial model checking and theorem proving for ensuring security properties, in: Proc. CSFW'98, IEEE Press, New York, 1998, pp. 44–52.
[22] F. Martinelli, Analysis of security protocols as open systems, Tech. Report IAT-B4-006, July 2001, accepted for publication on TCS (under minor revisions).
[23] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[24] P.Y.A. Ryan, S. Schneider, Process algebra and non-interference, in: Proc. CSFW'99, IEEE Press, New York, 1999, pp. 214–227.
[25] D. Sangiorgi, Expressing mobility in process algebra: first-order and higher-order paradigms, Ph.D. Thesis, University of Edimburgh, 1992.
[26] S. Schneider, Formal analysis of a non-repudiation protocol, in: Proc. CSFW'98, IEEE Press, New York, 1998, pp. 54–65.
[27] S. Schneider, Verifying authentication protocols in CSP, IEEE Trans. Software Eng. 24 (9) (1998).
[28] J.T. Wittbold, D.M. Johnson, Information flow in nondeterministic systems, in: Proc. 1990 IEEE Symp. on Research in Security and Privacy, IEEE Computer Society Press, Silver Spring, MD, 1990, pp. 144–161.