On Compositional Reasoning in the Spi-calculus*

Michele Boreale and Daniele Gorla

Dipartimento di Sistemi e Informatica, Università di Firenze
boreale@dsi.unifi.it , gorla@gdn.dsi.unifi.it

Abstract. Observational equivalences can be used to reason about the correctness of security protocols described in the spi-calculus. Unlike in CCS or in π-calculus, these equivalences do not enjoy a simple formulation in spi-calculus. The present paper aims at enriching the set of tools for reasoning on processes by providing a few equational laws for a sensible notion of spi-bisimilarity. We discuss the difficulties underlying compositional reasoning in spi-calculus and show that, in some cases and with some care, the proposed laws can be used to build compositional proofs. A selection of these laws forms the basis of a proof system that we show to be sound and complete for the strong version of bisimilarity.

Keywords: process calculi, axiomatization, reasoning on security

1 Introduction

Observational equivalences can be used to reason about the correctness of security protocols described in the spi-calculus [2]. Unlike in CCS or in π-calculus, these equivalences do not enjoy a simple formulation in spi-calculus. One reason is the interplay between cryptography and handling of private names, which somehow forces a distinction between the knowledge of the observer and the interface of the observed process (i.e. the set of its free names). On the contrary, in the simpler non-cryptographic setting, these two notions coincide. To illustrate this point, let us consider two processes that send different, encrypted messages along a public channel c: $P \triangleq (\nu\, k\,)\, \overline{c}\{a\}_k.\mathbf{0}$ and $Q \triangleq (\nu\, k\,)\, \overline{c}\{b\}_k.\mathbf{0}$. According to π-like bisimulation, P and Q are not equivalent, because they give rise to syntactically different transitions (names a and b are different). On the other hand, messages $\{a\}_k$ and $\{b\}_k$ have the same effect upon an external observer: in both cases, he/she simply cannot open the message because has no access to the private key k. Thus, it would be reasonable to regard P and Q as equivalent.

In [6], these considerations have led to the introduction of *environment sensitive (e.s.) bisimulation*, written $\sim$ below, where each process comes equipped with an explicit representation of what its environment knows. The latter is used to tell if two messages can be distinguished or not. Continuing with the example above, P and Q are e.s. bisimilar under the environment $\epsilon_{(a,b,c)}$ that accounts for the knowledge of the free names a, b and c: this is written as $(\ \epsilon_{(a,b,c)}\ ,\ \epsilon_{(a,b,c)}\ ) \vdash P \sim Q$. Environmental knowledge grows after an output

* Research partially supported by the Italian MURST Project NAPOLI and by the European FET project MIKADO (IST-2001-32222).

action is observed and two processes being compared may end up being placed in two different, though in some sense 'equivalent', environments. For instance, this is the case for P and Q above after the firing of the output actions. In [6], $\sim$ is shown to capture precisely barbed equivalence [12] in spi-calculus, which adds to the evidence that it is a sensible semantical notion.

The interplay between cryptography and private names makes compositional reasoning in spi-calculus difficult, when not impossible at all. A private name k can be extruded and hence become free, without this implying that k is learnt by the observer. Thus, we are sometimes confronted with equivalences like: $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \overline{c}\{a\}_k.P_1 \sim \overline{c}\{b\}_k.P_2$ where both σ_1 and σ_2 know a, b, c, but neither knows k. In general, this kind of equivalences are not preserved by parallel composition. For instance, when putting $\overline{c}k.\mathbf{0}$ in parallel to both sides of the previous relation, the equivalence breaks down. The reason is that $\overline{c}k.\mathbf{0}$ may provide the observer with the key k to open $\{a\}_k$ and $\{b\}_k$, thus enabling a distinction between these two messages. Similar problems arise from the non-deterministic choice and output prefix operators. (On the contrary, the congruence rules for restriction and input prefix appear to be somehow more liberal in spi-calculus than in π-calculus; this is further discussed in the paper).

In fact, one can devise congruence rules that work in some special cases and that are often sufficient to analyze concrete examples. In the paper, we show how to reason compositionally on a small security protocol like Wide Mouthed Frog [2]. However, special-case congruence rules appear to be not powerful enough to achieve completeness results in the same fashion as for CCS and π-calculus (see, e.g., [10, 13]). Indeed, proving any process equivalent to, say, a head normal formal (hnf) requires a congruence law for parallel composition that, as seen above, is not sound in general for $\sim$. We get round this difficulty by noting that the set of equations needed to re-write every spi-process to a hnf *are* preserved by parallel composition. Starting from this consideration, we design a two-level proof system. The first level only contains these general, hnf-related equations. The second level contains those identities that are specific to spi-calculus, plus a law for importing equalities from the first level. The resulting proof system is perhaps not as elegant as a one-level proof system might be, but provides a reasonably informative axiomatization of $\sim$ over finite processes.

For the sake of presentation, in this paper we confine ourselves to shared-key cryptography, as presented e.g. in [6]. We believe that, with obvious modifications, the same arguments apply when the language is extended with other crypto-primitives, like public-key encryption (see [2, 1]).

The rest of this paper is organized as follows. Sect. 2 provides a summary of the spi-calculus as presented in [6]. A set of equational laws that are meant to be useful in practice is presented in Sect. 3, together with some examples of their application. A more extensive example is in Sect. 4. Sect. 5 presents the proof system, and discuss its completeness. A few concluding remarks are in Sect. 6.

2 The Calculus and its Semantics

In this section we only define the basics of the calculus. We refer to [6] for a comprehensive presentation. The syntax of the calculus is summarized in Table 1.[1]

Table 1. Syntax of the calculus

$a, b \ldots, h, k, \ldots, x, y, z \ldots$	*names* $\mathcal{N}$
$M,\ N ::= a \mid \{M\}_k \mid \langle M_1, M_2 \rangle$	*messages* $\mathcal{M}$
$\eta,\ \zeta ::= a \mid \{\eta\}_\zeta \mid \mathtt{dec}_\eta(\zeta)$ $\mid \langle \eta, \zeta \rangle \mid \pi_1(\zeta) \mid \pi_2(\zeta)$	*expressions* $\mathcal{Z}$
$\phi,\ \psi ::= tt \mid name(\zeta) \mid [\zeta = \eta]$ $\mid \mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi \mid \phi \wedge \psi \mid \neg\phi$	*formulae* Φ
$P, Q ::= \mathbf{0} \mid \eta(x).P \mid \overline{\eta}\zeta.P \mid P + Q \mid P\,\vert\,Q$ $\mid\ !P \mid (\nu\, a\,)P \mid \phi P \mid \mathtt{let}\ z = \zeta\ \mathtt{in}\ P$	*processes* $\mathcal{P}$

It is assumed that $\mathtt{dec}_\cdot(\cdot)$ and $\pi_i(\cdot)$ do not occur in $name(\zeta)$, $[\zeta = \eta]$, $\eta(x)$ and $\overline{\eta}\zeta$. Operators $a(x).\cdot$, $(\nu\, a\,)\cdot$ and $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \cdot$ are *binders*, with the obvious scope, for names x, a and z, respectively. In $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \cdot$, it is assumed that $z \notin n(\zeta)$.

As usual, we use $\tilde{\cdot}$ to denote tuples of objects. Notions of alpha-equivalence, *free names* ($fn(P)$), *bound names* ($bn(P)$) and *names* ($n(P)$) of a process P arise as expected; in particular, we identify alpha-equivalent processes and formulae. We use two evaluation functions. The first one ($\widehat{\cdot} : \mathcal{Z} \to \mathcal{M} \cup \{\bot\}$) transforms an expression into a message by properly evaluating decryptions and projections (e.g. $\widehat{\mathtt{dec}_k(\{M\}_k)} = M$ while $\widehat{\mathtt{dec}_h(\{M\}_k)} = \bot$ if $h \neq k$). The second function ($[\![\cdot]\!] : \Phi \to \{tt, ff\}$) takes a formula and gives its boolean value as expected.

Informally, an *environment* represents the knowledge of names and keys that an external observer has acquired about a certain process. We represent and use an environment as a substitution σ of names with messages. The domain and proper co-domain of σ are written as $dom(\sigma)$ and $range(\sigma)$, respectively; we let $n(\sigma) \triangleq dom(\sigma) \cup (\cup_{M \in range(\sigma)} n(M))$. With ϵ_T we denote the substitution that acts like the identity on the set of names T. The extension of environment σ with the binding $[M/x]$, written $\sigma[M/x]$, and application σ to a term t, written $t\sigma$, are defined as usual.

A *configuration* is a pair $\sigma \triangleright P$ formed by an environment σ and a process P. Transitions between configurations take the form $\sigma \triangleright P \overset{\mu}{\underset{\delta}{\longmapsto}} \sigma' \triangleright P'$ and represent atomic interactions between P and σ. Here, μ is the *process action*

[1] Compared to the spi-calculus in [2], we have the traditional process operators (i.e. input and output prefixes, non-deterministic choice, parallel composition, replication and restriction) plus a more general form of boolean guard and an operator for decryption and pair splitting.

(i.e. input, output or τ) and δ is the complementary, under σ, *environment action* (respectively output, input and 'no action'). The inference rules in Table 2 mention in the premises early-style transitions of the form $P \xrightarrow{\mu} P'$: the latter just account for actions that processes are willing to do, regardless of whether an external observer can react or not to them. Its definition is standard and can be found in [6].

Table 2. Rules for the environment-sensitive calculus

It is assumed that $n(\eta) \subseteq dom(\sigma)$ and that names in $\widetilde{b}$ are fresh for σ and P.

$$\text{(E-OUT)}\ \frac{P \xrightarrow{(\nu\widetilde{b})\overline{a}\,\langle M\rangle} P' \qquad \widehat{\eta\sigma} = a}{\sigma \triangleright P \underset{\eta(x)}{\overset{(\nu\widetilde{b})\overline{a}\,\langle M\rangle}{\longmapsto}} \sigma[{}^{M}\!/\!_{x}] \triangleright P'} \qquad \text{(E-TAU)}\ \frac{P \xrightarrow{\tau} P'}{\sigma \triangleright P \underset{-}{\overset{\tau}{\longmapsto}} \sigma \triangleright P'}$$

$$\text{(E-INP)}\ \frac{P \xrightarrow{a\,M} P' \qquad \widehat{\eta\sigma} = a \qquad M = \widehat{\zeta\sigma} \qquad \widetilde{b} \triangleq (n(\zeta) - dom(\sigma))}{\sigma \triangleright P \underset{(\nu\widetilde{b})\overline{\eta}\,\langle\zeta\rangle}{\overset{a\,M}{\longmapsto}} \sigma[{}^{\widetilde{b}}\!/\!_{\widetilde{b}}] \triangleright P'}$$

2.1 Environment-Sensitive Strong Bisimulation

Definition 1 (Equivalence on environments). *Two environments σ_1 and σ_2 are* equivalent *(written $\sigma_1 \sim \sigma_2$) if $dom(\sigma_1) = dom(\sigma_2)$ and $\forall\ \phi$ s.t. $fn(\phi) \subseteq dom(\sigma_1)$ it holds $[\![\, \phi\sigma_1 \,]\!] = [\![\, \phi\sigma_2 \,]\!]$.*

For example, $\sigma_1 \triangleq [{}^{a}\!/\!_{x}\ ,\ {}^{b}\!/\!_{y}\ ,\ {}^{\{a\}_k}\!/\!_{z}]$ and $\sigma_2 \triangleq [{}^{a}\!/\!_{x}\ ,\ {}^{b}\!/\!_{y}\ ,\ {}^{\{b\}_k}\!/\!_{z}]$ are equivalent. On the contrary, $\sigma_1' \triangleq \sigma_1[{}^{k}\!/\!_{v}]$ and $\sigma_2' \triangleq \sigma_2[{}^{k}\!/\!_{v}]$ are not (consider the formula $\texttt{let}\ w = \texttt{dec}_v(z)\ \texttt{in}\ [x = w]$). A metatheory to reason on environment equivalence $\sim$ is presented in [6]; in particular, an equivalent definition is given there that leaves out universal quantifiers on ϕ and only works on the structure of the environments. However, the present definition suffices for our purposes.

If $\Re$ is a binary relation over configurations, we write $(\sigma_1 \triangleright P, \sigma_2 \triangleright Q) \in \Re$ as $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P\ \Re\ Q$; if $\sigma_1 \sim \sigma_2$ whenever $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P\ \Re\ Q$, we call $\Re$ *compatible*. We now give the strong version of e.s. bisimilarity from [6].

Definition 2 (Environment-sensitive strong bisimilarity). *Let $\Re$ be a binary and symmetric compatible relation of configurations. We say that $\Re$ is an* environment-sensitive strong bisimulation *if whenever $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P\ \Re\ Q$ and $\sigma_1 \triangleright P \underset{\delta}{\overset{\mu}{\longmapsto}} \sigma_1' \triangleright P'$ then there are μ', σ_2' and Q' such that $\sigma_2 \triangleright Q \underset{\delta}{\overset{\mu'}{\longmapsto}} \sigma_2' \triangleright Q'$ and $(\ \sigma_1'\ ,\ \sigma_2'\ ) \vdash P'\ \Re\ Q'$.* Environment-sensitive bisimilarity*, written $\sim$, is the largest environment-sensitive strong bisimulation.*

3 Laws for Compositional Reasoning

In this section we list and explain some laws that, in many cases, can be useful to prove bisimilarity of two processes in a simple and compositional way. As we shall see in Sect. 5, with some modifications, these laws form the basis of a complete proof system for $\sim$. In what follows, we discuss a little about significance of each law and provide explicative examples and countrexamples. Each law is a judgement of the form $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q$, where $\sigma_1 \sim \sigma_2$.

<u>*Basic laws :*</u> The following laws are easily derived from the definition.

$$\text{(REFL)} \quad (\ \sigma\ ,\ \sigma\ ) \vdash P \sim P \qquad \text{(SYM)} \quad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q}{(\ \sigma_2\ ,\ \sigma_1\ ) \vdash Q \sim P}$$

$$\text{(TRANS)} \quad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q \quad \wedge \quad (\ \sigma_2\ ,\ \sigma_3\ ) \vdash Q \sim R}{(\ \sigma_1\ ,\ \sigma_3\ ) \vdash P \sim R}$$

We then have a form of 'weakening' which states that discarding some entries from the environments does preserve equivalence.

$$\text{(WEAK)} \quad \frac{(\ \sigma_1[\widetilde{M_1}/\widetilde{x}]\ ,\ \sigma_2[\widetilde{M_2}/\widetilde{x}]\ ) \vdash P \sim Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q}$$

The following law provides a basic step in compositional reasoning. Note that the more general formulation "$(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim P \quad if\ \sigma_1 \sim \sigma_2$" does not hold.

$$\text{(NIL)} \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash \mathbf{0} \sim \mathbf{0} \qquad \text{if } \sigma_1 \sim \sigma_2$$

Example 1. In fact, let us consider $P \triangleq p(x).\mathtt{let}\ y = \mathtt{dec}_k(x)\ \mathtt{in}\ [y = a]\overline{p}y.\mathbf{0}$ and the equivalent environments $\sigma_1 \triangleq [p/p\ ,\ \{a\}_k/y]$ and $\sigma_2 \triangleq [p/p\ ,\ \{b\}_k/y]$. A transition from $\sigma_1 \triangleright P$ with $\delta = \overline{p}y$ leads to a process that is capable of a $\overline{p}a$-action, while the corresponding transition from $\sigma_2 \triangleright P$ leads to a process that is stuck, because the matching $[b = a]$ is not satisfied. Thus $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \not\sim P$. $\square$

The standard scope extrusion law can also be useful:

$$\text{(EXTR)} \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash (\nu\, k\,)\,(P\,|\,Q) \sim ((\nu\, k\,)\,P)\,|\,Q \qquad \text{if } k \notin fn(Q)$$

The last basic law can be used to replace a decryption underneath a `let` with an equality test, under some conditions. Here and in the rest of this paper, we will use the abbreviation "$a(\{y\}_k).P$" for "$a(x).\mathtt{let}\ y = \mathtt{dec}_k(x)\ \mathtt{in}\ P$", where $x \notin fn(P,k)$. In the following law we also use the notion of *context* that is a generic process with a 'hole' that can be filled by any process.

(LET$_1$) $\quad (\ \sigma\ ,\ \sigma\ ) \vdash (\nu\, \widetilde{h}, k\,)\,(\ C[\ \overline{p}\{M\}_k.P\ ]\ |\ D[\ p(\{y\}_k).Q\ ])\ \sim$
$\qquad\qquad (\nu\, \widetilde{h}, k\,)\,(\ C[\ \overline{p}\{M\}_k.P\ ]\ |\ D[\ p(x).[x = \{M\}_k]Q[M/y]\ ])$

$k \notin n(P, C, D)$, contexts C and D do not bind names in $\overline{p}\{M\}_k$ and x is fresh

Example 2. In order to better understand the above law, consider the process $P \triangleq (\nu\, k\,)\,(A\,|\,B)$ where $A \triangleq \overline{p}\{M\}_k.R$, $B \triangleq p(\{y\}_k).Q$ and $k \notin n(R)$. Intuitively, since k is initially not known by the environment, the only message B can read and then decrypt using k is the one written by A, i.e. $\{M\}_k$. In fact, the condition $k \notin n(R)$ prevents the environment from learning k, or anything encrypted with it, before Q evolves. Thus P is equivalent to $(\nu\, k\,)\,(A\,|\,p(x).[x = \{M\}_k]Q[M/y])$, where x is a fresh variable. $\square$

Output congruence :

$$\text{(OUT)}\qquad \frac{(\ \sigma_1[M_1/x]\ ,\ \sigma_2[M_2/x]\ ) \vdash P \sim Q}{(\ \sigma_1[M_1/x]\ ,\ \sigma_2[M_2/x]\ ) \vdash \overline{a_1}M_1.P \sim \overline{a_2}M_2.Q}$$

where $a_i\ =\ \widehat{\eta\sigma_i}$ for $i = 1, 2$ and for some η *s.t.* $n(\eta) \subseteq dom(\sigma_1)$

Notice that the two channels a_1 and a_2 may be different but are related via the two environments (e.g. a_1 and a_2 may be stored in the same entry of the two environments). Similarly, the messages M_1 and M_2 may well be different, but they must correspond to the same environment entry x. The use of (OUT) is tipically joined with the use of (WEAK), as shown below.

Example 3. We want to prove $(\ \sigma\ ,\ \sigma\ ) \vdash P \sim Q$, where $\sigma \triangleq [p/p]$, $P \triangleq \overline{p}\{a\}_k.\mathbf{0}$ and $Q \triangleq \overline{p}\{b\}_k.\mathbf{0}$. Notice that $\sigma[\{a\}_k/x] \sim \sigma[\{b\}_k/x]$, since neither of $\{a\}_k$ and $\{b\}_k$ can be opened using the knowledge in σ. So, by (NIL), we get $(\ \sigma[\{a\}_k/x]\ ,\ \sigma[\{b\}_k/x]\ ) \vdash \mathbf{0} \sim \mathbf{0}$. Then, by (OUT) and (WEAK) (discarding the x entry), we conclude. $\square$

Input congruence : In the rest of the paper we shall use the following predicate:

$$\Gamma(\ \zeta,\ \widetilde{b},\ \sigma_1,\ \sigma_2,\ P,\ Q\ )\ \ \stackrel{\triangle}{\Leftrightarrow}\ \ \widehat{\zeta\sigma_1} \neq \bot\ \wedge\ \widetilde{b} = n(\zeta) - dom(\sigma_1)\ \wedge\ \widetilde{b}\ \cap n(\sigma_1, \sigma_2, P, Q) = \emptyset$$

(notice that $\Gamma(\ \zeta,\ \widetilde{b},\ \sigma_1,\ \sigma_2,\ P,\ Q\ ) \Leftrightarrow \Gamma(\ \zeta,\ \widetilde{b},\ \sigma_2,\ \sigma_1,\ P,\ Q\ )$, if $\sigma_1 \sim \sigma_2$).

$$\text{(IN)}\qquad \frac{\forall\ \zeta, \widetilde{b}\ s.t.\ \ \Gamma(\ \zeta,\ \widetilde{b},\ \sigma_1,\ \sigma_2,\ P,\ Q\ )\ :\quad (\ \sigma_1[\widetilde{b}/\widetilde{b}]\ ,\ \sigma_2[\widetilde{b}/\widetilde{b}]\ ) \vdash P[\widehat{\zeta\sigma_1}/x] \sim Q[\widehat{\zeta\sigma_2}/x]}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash a_1(x).P \sim a_2(x).Q}$$

where $a_i\ =\ \widehat{\eta\sigma_i}$ for $i = 1, 2$ and for some η *s.t.* $n(\eta) \subseteq dom(\sigma_1)$

The above formulation of the input rule is somehow more generous than the π-calculus style one. In fact, the premise does not require instantiating x to all possible messages, but only to those that can be built out of σ_1, σ_2 and some new names $\widetilde{b}$, as specified by all ζ's that satisfies the predicate Γ. This is made clearer in the following example.

Example 4. Using (IN), we can prove an equivalence quite similar to (LET$_1$) (in the sense that a "`let`" is replaced with a test), that is

$$(\ \sigma\ ,\ \sigma\ ) \vdash a(x).\mathtt{let}\ x' = \mathtt{dec}_k(x)\ \mathtt{in}\ P \sim a(x).[x = \{b\}_k]P[{}^b/_{x'}]$$

where $\sigma \triangleq [{}^a/_a\ ,\ {}^{\{b\}_k}/_w]$. Indeed, it is easy to check that, whenever $\Gamma(\ \zeta,\ \widetilde{c},\ \sigma,\ \sigma,\ \mathtt{let}\ldots,\ [x = \ldots\ )$ and $\widehat{\zeta\sigma} = \{M\}_k$, then $M = b$. This is a consequence of the fact that the environment does not know k and $k \notin \widetilde{c}$. On the contrary, if the environment knew k, it could have created the message $\{a\}_k$, which, upon reception, would have stopped the second process and not the first one. Of course, the above two processes are not equivalent according to traditional bisimilarity. □

The formulation of the input rule given above fails to capture some equivalences: in fact, in the proof system, it will be replaced by a more general rule.

Example 5. Let us consider the following $\sigma \triangleq [{}^a/_a]$, $P \triangleq a(x).\overline{a}\{x\}_k.\mathbf{0}$ and $Q \triangleq a(x).[x \neq b]\overline{a}\{x\}_k.\mathbf{0} + a(x).\overline{a}\{b\}_k.\mathbf{0}$. It is easy to see that $(\ \sigma\ ,\ \sigma\ ) \vdash P \sim Q$ but this equivalence is not an instance of (IN). □

Parallel composition :

$$\text{(PAR)}\qquad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \mid R\sigma_1 \sim Q \mid R\sigma_2}\ fn(R) \subseteq dom(\sigma_1)$$

As pointed out in the introduction, congruence under parallel composition is a major problem in spi-calculus. In fact, a naive formulation like

$$\frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q \quad \wedge \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash R \sim S}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \mid R \sim Q \mid S}$$

is not valid.

Example 6. To see this, let us consider $P \triangleq \overline{p}\{a\}_k.\mathbf{0}$, $Q \triangleq \overline{p}\{b\}_k.\mathbf{0}$, $R \triangleq \overline{p}k.\mathbf{0}$ and $\sigma \triangleq [{}^p/_p\ ,\ {}^a/_a\ ,\ {}^b/_b]$. Following Example 3, we can prove $(\ \sigma\ ,\ \sigma\ ) \vdash P \sim Q$. However, $(\ \sigma\ ,\ \sigma\ ) \vdash\ P \mid R \not\sim Q \mid R$. In fact the output of the key k enables an external observer to distinguish $\{a\}_k$ from $\{b\}_k$, hence $P \mid R$ from $Q \mid R$. □

The side condition of (PAR) reduces the set of processes that can be composed with P and Q, by requiring that the composed processes are consistent with the knowledge available to the environment. In spite of this limitation, the rule allows for non trivial forms of compositional reasoning, as shown in Sect. 4.

Other Congruences :

$$(\textsc{Sum})\quad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_1 \sim P_2 \quad \wedge \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash Q_1 \sim Q_2}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_1 + Q_1 \sim P_2 + Q_2}$$

$$(\textsc{Res})\quad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash (\nu\, \widetilde{h}_1\,)\, P \sim (\nu\, \widetilde{h}_2\,)\, Q}\ \widetilde{h}_i \cap n(\sigma_i) = \emptyset \ for\ i = 1, 2$$

$$(\textsc{Guard})\quad \frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \phi P \sim \phi Q}$$

$$(\textsc{Let}_2)\quad \frac{\widehat{\zeta} \neq \bot \quad \wedge \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash P[\widehat{\zeta}/z] \sim Q[\widehat{\zeta}/z]}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \mathtt{let}\ z = \zeta\ \mathtt{in}\ P \ \sim\ \mathtt{let}\ z = \zeta\ \mathtt{in}\ Q}$$

The only surprising aspect of these laws is in (Res): the tuples of restricted names $\widetilde{h}_1$ and $\widetilde{h}_2$ can be different even in length.

Example 7. Using (Res) one can prove $(\ \sigma\ ,\ \sigma\ ) \vdash P \sim (\nu\, k\,)\, P$, provided that $k \notin n(\sigma)$. □

4 A Compositional Secrecy Proof for WMF

In this section we verify a property of the *Wide Mouthed Frog* (WMF) protocol. Differently from [2, 6], our proof is entirely compositional and syntactical, based on the simple equational laws introduced in the previous section.

Consider a system where two agents A and B share two secret keys, k_{AS} and k_{BS} respectively, with a server S. The purpose of the protocol is to establish a new secret key k between A and B, which A may use to pass some confidential information d to B. We suppose that the protocol is always started by A and that all communications occur on a public channel, say p. Informally, the protocol can be described as follows:

Message 1	$A \rightarrow S:$	$\{k\}_{k_{AS}}$
Message 2	$S \rightarrow B:$	$\{k\}_{k_{BS}}$
Message 3	$A \rightarrow B:$	$\{d\}_k$.

Our intent here is to verify a secrecy property for one run of the protocol. In our language, the above notation translates to a process $P(d)$ defined, like in [2], as follows (we use the notation $t(w)$ to emphasize that name w may occur free in t and, for any M, $t(M)$ abbreviates $t[M/w]$; bound names are all distinct):

$$\begin{aligned} A(d) &\stackrel{\triangle}{=} \overline{p}\{k\}_{k_{AS}}.\overline{p}\{d\}_k.G \\ S &\stackrel{\triangle}{=} p(\{x\}_{k_{AS}}).\overline{p}\{x\}_{k_{BS}}.\mathbf{0} \\ B &\stackrel{\triangle}{=} p(\{y\}_{k_{BS}}).p(\{z\}_y).F(z) \\ P(d) &\stackrel{\triangle}{=} (\nu\, k_{AS},\ k_{BS}\,)\,(((\nu\, k\,)\, A(d))\,|\,S\,|\,B)\ . \end{aligned}$$

The processes G and $F(z)$ represent behaviuor of A and B respectively upon completion of the protocol. Below, we make the following assumptions:

1. the key k does not occur in G and F (that is, k is one-time);
2. k_{AS} and k_{BS} are used only to establish the new session key (in particular $k_{AS}, k_{BS} \notin n(F(z), G)$);
3. for each M_1, $M_2 \in \mathcal{M}$ and for each $\sigma_1(w)$ and $\sigma_2(w)$ s.t. $\sigma_1(M_1) \sim \sigma_2(M_2)$, it holds $(\ \sigma_1(M_1)\ ,\ \sigma_2(M_2)\ ) \vdash F(M_1) \sim F(M_2)$.

Assumption 1 is necessary in order to apply (Par), while assumtion 2 is necessary to apply (Let$_1$). At the moment we do not know how to discard these assumptions while preserving the compositionality of the proof. Assumption 3 seems reasonable because F should not leak the received datum itself. Following [2], the desired secrecy property is

$$\text{“}P(d)\ \textit{does not leak}\ d\text{”} : \forall\ M, M' \in \mathcal{M}\quad (\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M) \sim P(M')\quad \text{where}\quad V \triangleq fn(P(M), P(M'))$$

The proof of the above assertion takes these four steps (by alpha-equivalence, we assume $k_{AS}, k_{BS}, k \notin n(M, M')$):

(i) By (Extr), $(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M) \sim (\nu\, k_{AS},\ k_{BS},\ k\,)\,(A(M) \,|\, S \,|\, B)$. Then, by (Let$_1$) (applied to A and S) and (Trans), $(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M) \sim (\nu\, k_{AS},\ k_{BS},\ k\,)\,(A(M) \,|\, S' \,|\, B)$ where $S' \triangleq p(u).[u = \{k\}_{k_{AS}}]\overline{p}\{k\}_{k_{BS}}.\mathbf{0}$. Again by (Let$_1$) (applied to S' and B) and (Trans), we obtain

$$(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M) \sim (\nu\, k_{AS},\ k_{BS},\ k\,)\,(A(M) \,|\, S' \,|\, B') \tag{1}$$

where $B' \triangleq p(v).[v = \{k\}_{k_{BS}}]p(w).\mathtt{let}\ z = \mathtt{dec}_k(w)\ \mathtt{in}\ F(z)$.

(ii) Similarly, replacing M with M', we obtain

$$(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M') \sim (\nu\, k_{AS},\ k_{BS},\ k\,)\,(A(M') \,|\, S' \,|\, B')\ . \tag{2}$$

(iii) Let us define the environments $\sigma \triangleq \epsilon_V[\{k\}_{k_{AS}}/x_1\ ,\ \{k\}_{k_{BS}}/x_2\ ,\ \{M\}_k/x_3]$ and $\sigma' \triangleq \epsilon_V[\{k\}_{k_{AS}}/x_1\ ,\ \{k\}_{k_{BS}}/x_2\ ,\ \{M'\}_k/x_3]$, that are equivalent. Now suppose that we can prove

$$(\ \sigma\ ,\ \sigma'\ ) \vdash B' \sim B'\ . \tag{3}$$

We let $S'' \triangleq p(u).[u = x_1]\overline{p}x_2.\mathbf{0}$ and $A'' \triangleq \overline{p}x_1.\overline{p}x_3.G$. We trivially have $S' = S''\sigma = S''\sigma'$, $A(M) = A''\sigma$ and $A(M') = A''\sigma'$. Thus, by (Par) and (3), we have $(\ \sigma\ ,\ \sigma'\ ) \vdash (A'' \,|\, S'')\sigma \,|\, B' \sim (A'' \,|\, S'')\sigma' \,|\, B'$ which is the same as $(\ \sigma\ ,\ \sigma'\ ) \vdash A(M) \,|\, S' \,|\, B' \sim A(M') \,|\, S' \,|\, B'$. By (Weak), we obtain $(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash A(M) \,|\, S' \,|\, B' \sim A(M') \,|\, S' \,|\, B'$ and by (Res) we have $(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash (\nu\, k_{AS}, k_{BS}, k\,)\,(A(M) \,|\, S' \,|\, B') \sim (\nu\, k_{AS}, k_{BS}, k\,)\,(A(M') \,|\, S' \,|\, B')$. This equation, together with (Trans), (1) and (2), allows us to conclude the desired $(\ \epsilon_V\ ,\ \epsilon_V\ ) \vdash P(M) \sim P(M')$.

(iv) We are now left with proving (3). The following steps prove our claim.
   1. $(\ \sigma\ ,\ \sigma'\ ) \vdash F(M) \sim F(M')$

2. $(\ \sigma\ ,\ \sigma'\ ) \vdash \texttt{let}\ z = \texttt{dec}_k(\{M\}_k)\ \texttt{in}\ F(z) \sim$
$\texttt{let}\ z = \texttt{dec}_k(\{M'\}_k)\ \texttt{in}\ F(z)$
3. $(\ \sigma\ ,\ \sigma'\ ) \vdash p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z) \sim$
$p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z)$
4. $(\ \sigma\ ,\ \sigma'\ ) \vdash [v = \{k\}_{k_{BS}}]p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z) \sim$
$[v = \{k\}_{k_{BS}}]p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z)$
5. $(\ \sigma\ ,\ \sigma'\ ) \vdash p(v).[v = \{k\}_{k_{BS}}]p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z) \sim$
$p(v).[v = \{k\}_{k_{BS}}]p(w).\texttt{let}\ z = \texttt{dec}_k(w)\ \texttt{in}\ F(z)$
6. $(\ \sigma\ ,\ \sigma'\ ) \vdash B' \sim B'$

where: step 1 follows by assumption, step 2 by (Let$_2$), step 3 by (In),[2] step 4 by (Guard), step 5 by (In) (with considerations similar to 3) and finally step 6 by definition of B'.

Before leaving this example, we would like to stress that, in a similar way, we can prove other properties of the protocol, like integrity (*"if B accepts a message $\{N\}_k$ then $N = d$"*) and key authentication (*"B accepts only the key k generated by A"*). Moreover, following [7], we have also applied the same steps to a simplified Kerberos protocol, obtaining similar results; the work was only complicated by the presence of tuples of messages. We have presented WMF for its readability.

5 A Proof System

In this section we present a sound and complete proof system for $\sim$ over finite processes; we leave out from our language the replication operator ('!') which would lead to an undecidable theory (as shown in [11] for the π-calculus). The proof system has two levels. The first level is a proof system sound for ordinary strong bisimilarity (in the same vein as, e.g., the classical ones of the π-calculus [13]). The second level is tailored to the environment-sensitive semantics. Since ordinary bisimulation is finer than e.s. bisimulation, equivalences proven within the first proof system can be imported to the second one.

5.1 The Proof System *S1*

Definition 3. *A* strong bisimulation *is a symmetric relation between processes $\Re$ s.t. whenever $P \ \Re\ Q$ and $P \xrightarrow{\mu} P'$ then there is Q' s.t. $Q \xrightarrow{\mu} Q'$ and $P' \ \Re\ Q'$. We call* bisimilarity *(written $\dot\sim$) the largest strong bisimulation.*

Table 3 displays the axioms for $\dot=$. Essentially these are the axioms for bisimilarity in the π-calculus (see for example [13]) with some additional laws

[2] It is easy to check that for each ζ and $\widetilde{b}$ s.t. $\Gamma(\ \zeta,\ \widetilde{b},\ \sigma,\ \sigma',\ \texttt{let}\ldots,\ \texttt{let}\ldots\ )$, if $\widehat{\zeta\sigma} = \{N\}_k$ and $\widehat{\zeta\sigma'} = \{N'\}_k$, then $N = M$ and $N' = M'$. In other words, the only readable message of the form $\{\cdot\}_k$ is that contained in x_3 and by assumption $(\ \sigma\ ,\ \sigma'\ ) \vdash F(M) \sim F(M')$. If $\widehat{\zeta\sigma}$ is not of the form $\{\cdot\}_k$, then both processes are stuck, thus trivially equivalent to $\mathbf{0}$.

Table 3. The proof system *S1*

Axioms :

Monoid axioms for $+$ *and* $\mathbf{0}$

Monoid axioms for $|$ *and* $\mathbf{0}$

(Abs) $\quad P + P \doteq P$

(Exp) $$\sum_{i\in I}\alpha_i.P_i \,|\, \sum_{j\in J}\beta_j.Q_j \;\doteq\; \begin{array}{l} \sum_{i\in I}\alpha_i.(P_i \,|\, \sum_{j\in J}\beta_j.Q_j) + \\ \sum_{j\in J}\beta_j.(\sum_{i\in I}\alpha_i.P_i \,|\, Q_j) + \\ \sum^{\alpha_i \,=\, a_i(x)}_{\beta_j \,=\, (\nu\,\widetilde{b_j})\,\overline{a_i}M_j} \tau.(\nu\,\widetilde{b_j})\,(P_i[M_j/x] \,|\, Q_j) + \\ \sum^{\beta_j \,=\, a_j(x)}_{\alpha_i \,=\, (\nu\,\widetilde{c_i})\,\overline{a_j}N_i} \tau.(\nu\,\widetilde{c_i})\,(P_i \,|\, Q_j[N_i/x]) \end{array}$$

(Res$_1$) $\quad (\nu\, n)\, P \doteq P \qquad$ if $n \notin fn(P)$

(Res$_2$) $$(\nu\, n)\sum_{i\in I}\alpha_i.P_i \;\doteq\; \begin{array}{l} \sum^{i\in I}_{n\notin n(\alpha_i)}\alpha_i.(\nu\, n)\, P_i + \\ \sum^{i\in I}_{\alpha_i \,=\, (\nu\,\widetilde{b})\,\overline{a}M \,:\, n\neq a \,\wedge\, n\in n(M)} ((\nu\, n)\,\alpha_i).P_i \end{array}$$

(Phi) $$\phi P \;\doteq\; \begin{cases} \mathbf{0} & \text{if } [\![\,\phi\,]\!] = \mathit{ff} \\ P & \text{otherwise} \end{cases}$$

(Let$_3$) $$\texttt{let } z = \zeta \texttt{ in } P \;\doteq\; \begin{cases} \mathbf{0} & \text{if } \widehat{\zeta} = \bot \\ P[\widehat{\zeta}/z] & \text{otherwise} \end{cases}$$

Congruence laws :

Congruence laws for $+$, $|$ *and* ν

(Phi$_0$) $$\frac{P \doteq Q}{\phi P \doteq \phi Q}$$

(Let$_0$) $$\frac{P[\widehat{\zeta}/z] \doteq Q[\widehat{\zeta}/z]}{\texttt{let } z = \zeta \texttt{ in } P \doteq \texttt{let } z = \zeta \texttt{ in } Q} \quad \widehat{\zeta} \neq \bot$$

specific to the spi-calculus. Note that (LET_0) can be derived from (LET_3); we keep both for the sake of uniformity.

Proposition 1. *If $P \doteq Q$ then $P \dot{\sim} Q$.*

In what follows, we use the notion of *bound output prefix*, that is an output prefix with some restrictions of the form $(\nu \widetilde{b})\,\overline{a}M$, where $a \notin \widetilde{b}$ and $\widetilde{b} \subseteq n(M)$.

Definition 4. *A process P is in* head normal form *(written hnf) if it is of the form $\sum_{i \in I} \alpha_i.P_i$ where α_i is a generic input-, (bound) output- or τ-prefix.*

Let us denote by $|\,P\,|$ the *depth* of a process P, inductively defined by

$$|\,P\,| \triangleq \begin{cases} 0 & \text{if } P = \mathbf{0} \\ 1 + |\,Q\,| & \text{if } P = \alpha.Q \\ max\{|\,Q\,|, |\,R\,|\} & \text{if } P = Q + R \\ |\,Q\,| + |\,R\,| & \text{if } P = Q \,|\, R \\ |\,Q\,| & \text{if } P = (\nu\, n\,)\,Q\ ,\ P = \phi Q \text{ or } P = (\mathtt{let}\ z = \zeta\ \mathtt{in}\ Q)\ . \end{cases}$$

Lemma 1. *For each process P there is a hnf P' s.t. $|\,P'\,| \leq |\,P\,|$ and $P \doteq P'$.*

The proof is by a standard induction over the number of operators in P. The inductive case for $P = P_1 \,|\, P_2$ relies on π-like congruence of $\dot{\sim}$ for '$|$'.

5.2 The Proof System *S2*

The proof system is presented in Table 4. It consists of a selection (and modification) of laws presented in Sect. 3, plus the rule (IMPORT). The latter can be used to import equalities proved within *S1* into *S2*. Also notice the more general form of the input congruence, akin to those found in [9, 5].

Theorem 1 (Soundness). *If $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q$ then $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q$.*

The main step of the completeness proof is that *S2* is complete for finite hnf (then, using (IMPORT), the proof for general finite processes is immediate). In order to obtain this first result we need a simple lemma:

Lemma 2. *For each $a \in \mathcal{N}$ let $a^{-1}_{\sigma_1} \triangleq \{\eta \in \mathcal{Z} : n(\eta) \subseteq dom(\sigma_1)\ \wedge\ \widehat{\eta\sigma_1} = a\}$. If $\sigma_1 \sim \sigma_2$, then there is a unique name b s.t. $\widehat{\eta\sigma_2} = b$ for each $\eta \in a^{-1}_{\sigma_1}$.*

Proposition 2 (Completeness for hnf). *Let P and Q be finite hnf. If $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q$ then $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q$.*

Proof. By induction on $|\,P\,| + |\,Q\,|$. The base case is trivial using (NIL). For the inductive step, we group the summands by the kind of their prefixes, obtaining $P \triangleq \sum_I \alpha_i.P_i = P_\tau + P_{out} + P_{in}$ and $Q \triangleq \sum_J \beta_j.Q_j = Q_\tau + Q_{out} + Q_{in}$. It is sufficient to prove that $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_s = Q_s$ for $s \in \{\ \tau\ ,\ out\ ,\ in\ \}$.

$s = \tau$. We will prove that each summand of P_τ is provably equal to a summand of Q_τ. Consider $\sigma_1\ \triangleright\ P \overset{\tau}{\underset{-}{\longmapsto}} \sigma_1\ \triangleright\ P_i$. By hypothesis, there is $j \in J$ s.t.

Table 4. The proof system *S2*

Axioms :

(Nil) $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \mathbf{0} = \mathbf{0}$ if $\sigma_1 \sim \sigma_2$

(Refl) $(\ \sigma_1\ ,\ \sigma_1\ ) \vdash P = P$

(Import) $$\frac{P\ \doteq\ Q}{(\ \sigma\ ,\ \sigma\ ) \vdash P = Q}$$

(Sym) $$\frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q}{(\ \sigma_2\ ,\ \sigma_1\ ) \vdash Q = P}$$

(Weak) $$\frac{(\ \sigma_1[\widetilde{M}_1/\widetilde{x}]\ ,\ \sigma_2[\widetilde{M}_2/\widetilde{x}]\ ) \vdash P = Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q}$$

(Trans) $$\frac{\begin{array}{c}(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q\ \wedge \\ (\ \sigma_2\ ,\ \sigma_3\ ) \vdash Q = R\end{array}}{(\ \sigma_1\ ,\ \sigma_3\ ) \vdash P = R}$$

Congruence laws :

(Tau) $$\frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \tau.P = \tau.Q}$$

(Out) $$\frac{(\ \sigma_1[M_1/x]\ ,\ \sigma_2[M_2/x]\ ) \vdash P = Q}{(\ \sigma_1[M_1/x]\ ,\ \sigma_2[M_2/x]\ ) \vdash \overline{a_1}M_1.P = \overline{a_2}M_2.Q}$$

where $a_i\ =\ \widehat{\eta\sigma_i}$ for $i = 1, 2$ and for some η $s.t.$ $n(\eta) \subseteq dom(\sigma_1)$

(Inp) $$\frac{\begin{array}{l}\forall\ \zeta, \widetilde{b}\ s.t.\ \Gamma(\ \zeta,\ \widetilde{b},\ \sigma_1,\ \sigma_2,\ \sum_{i\in I} P_i,\ \sum_{j\in J} Q_j\ )\quad : \\ \qquad (\ \sigma_1[\widetilde{b}/\widetilde{b}]\ ,\ \sigma_2[\widetilde{b}/\widetilde{b}]\ ) \vdash \sum_{i\in I} \tau.P_i[\widehat{\zeta\sigma_1}/x] = \sum_{j\in J} \tau.Q_j[\widehat{\zeta\sigma_2}/x]\end{array}}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \sum_{i\in I} a_1(x).P_i = \sum_{j\in J} a_2(x).Q_j}$$

where $a_i\ =\ \widehat{\eta\sigma_i}$ for $i = 1, 2$ and for some η $s.t.$ $n(\eta) \subseteq dom(\sigma_1)$

(Sum) $$\frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_1 = P_2 \quad \wedge \quad (\ \sigma_1\ ,\ \sigma_2\ ) \vdash Q_1 = Q_2}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_1 + Q_1 = P_2 + Q_2}$$

(Res) $$\frac{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q}{(\ \sigma_1\ ,\ \sigma_2\ ) \vdash (\nu\,\widetilde{h}_1\,)\,P \sim (\nu\,\widetilde{h}_2\,)\,Q}\ \widetilde{h}_i \cap n(\sigma_i) = \emptyset\ for\ i = 1, 2$$

$\sigma_2 \triangleright Q \overset{\tau}{\underset{-}{\longmapsto}} \sigma_2 \triangleright Q_j$ and $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_i \sim Q_j$. We cannot apply induction because P_i and Q_j are in general not hnf. However, by Lemma 1 and (IMPORT), we can find a hnf P' such that $(\ \sigma_1\ ,\ \sigma_1\ ) \vdash P_i\ = P'$ and $|\, P'\,| \leq |\, P_i\,|$ (and similarly for Q_j and Q'). By Theorem 1 and (TRANS), we obtain $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P' \sim Q'$. By induction and (TRANS), we obtain $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_i = Q_j$ and thus, by (TAU), $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \tau.P_i = \tau.Q_j$. Repeating this for each summand of P_τ, then for each summand of Q_τ and finally summing up (using (ABS) if necessary), we have the desired $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_\tau = Q_\tau$.

$s = out$. Similar to the previous case.

$s = in$. Let us partition the summands of P_{in} and Q_{in} according to their input channels, that is, let us write $P_{in} = P_{a_1} + \ldots + P_{a_m}$ and $Q_{in} = Q_{b_1} + \ldots + Q_{b_m}$. Let us consider now P_{a_1} and let us pick up any η s.t. $\widehat{\eta\sigma_1} = a_1$. By Lemma 2 we know that, indipendently from our choice of η, there is a unique $h \in \{1, \ldots, m\}$ s.t. $\widehat{\eta\sigma_2} = b_h$. We want to prove that $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_{a_1} = Q_{b_h}$. We define the sets $I' \stackrel{\triangle}{=} \{i \in I : \alpha_i = a_1(x)\}$ and $J' \stackrel{\triangle}{=} \{j \in J : \beta_j = b_h(x)\}$ and consider a generic ζ and $\widetilde{c}$ such that $\Gamma(\ \zeta,\ \widetilde{c},\ \sigma_1,\ \sigma_2,\ \sum_{I'} P_i,\ \sum_{J'} Q_j\ )$; moreover, let $M_i \stackrel{\triangle}{=} \widehat{\zeta\sigma_i}$, for $i = 1, 2$. By bisimilarity hypotesis and Lemma 2, for any $i \in I'$, there is a $j_i \in J'$ s.t. $(\ \sigma_1[\widetilde{c}/\widetilde{c}]\ ,\ \sigma_2[\widetilde{c}/\widetilde{c}]\ ) \vdash P_i[M_1/x] \sim Q_{j_i}[M_2/x]$. Similarly to τ-case, we can prove that $(\ \sigma_1[\widetilde{c}/\widetilde{c}]\ ,\ \sigma_2[\widetilde{c}/\widetilde{c}]\ ) \vdash \tau.P_i[M_1/x] = \tau.Q_{j_i}[M_2/x]$. Repeating this for each $i \in I'$ and finally summing up these equalities (using (ABS) if necessary) we have

$$(\ \sigma_1[\widetilde{c}/\widetilde{c}]\ ,\ \sigma_2[\widetilde{c}/\widetilde{c}]\ ) \vdash \sum_{i \in I'} \tau.P_i[M_1/x] = \sum_{i \in I'} \tau.Q_{j_i}[M_2/x]\ . \tag{4}$$

Similarly, for each $j \in J'$, we can find $i_j \in I'$ s.t.

$$(\ \sigma_1[\widetilde{c}/\widetilde{c}]\ ,\ \sigma_2[\widetilde{c}/\widetilde{c}]\ ) \vdash \sum_{j \in J'} \tau.P_{i_j}[M_1/x] = \sum_{j \in J'} \tau.Q_j[M_2/x]\ . \tag{5}$$

Now, notice that the left hand side of (5) is a sub-summatory of the left hand side of (4), and symmetrically for the right hand sides. Thus, by (SUM) and (ABS), we obtain $(\ \sigma_1[\widetilde{c}/\widetilde{c}]\ ,\ \sigma_2[\widetilde{c}/\widetilde{c}]\ ) \vdash \sum_{i \in I'} \tau.P_i[M_1/x] = \sum_{j \in J'} \tau.Q_j[M_2/x]$. We can apply (INP) and obtain $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash \sum_{i \in I'} a_1(x).P_i = \sum_{j \in J'} b_h(x).Q_j$, that is, $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_{a_1} = Q_{b_h}$. Repeating this for each summand of P_{in}, then for each summand of Q_{in} and finally summing up (using (ABS) if necessary), we obtain $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P_{in} = Q_{in}$. $\square$

Theorem 2 (Completeness). *Let P and Q be finite processes. If $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P \sim Q$ then $(\ \sigma_1\ ,\ \sigma_2\ ) \vdash P = Q$.*

6 Conclusions and Related Work

We have presented a set of equational laws for the spi-calculus that is useful to reason on concrete examples and forms the basis for a complete two-level proof system.

Our work can be extended in several directions. For example, one might consider the weak version of e.s. bisimilarity and/or a language extended with different crypto-primitives (in both cases, we do not foresee serious conceptual obstacles, though). Another direction is related to finding equational rules for the replication operator, which could be useful when, e.g., reasoning on protocols with an unbounded number of participants. A problem left open by our presentation is how to reduce to a finitary form the input congruence rule, which contains an infinitary premise. We think that, at least in the case of finite processes, bounds on the size of the ζ's can be statically determined. Symbolic techniques, in the same vein as [9, 5, 4], could be helpful (See also [8]).

The spi-calculus was introduced by Abadi and Gordon in [2]. Early work on spi-bisimilarity was presented in [3], where *framed* bisimulation was presented as a proof technique, though incomplete, for reasoning on contextual equivalences. In [6], e.s. bisimilarity was introduced and proved to be a (purely coinductive) characterization of barbed equivalence [12] in spi-calculus. Some of the congruence rules used in this paper were introduced there, but no proof system was defined.

References

1. M. Abadi, C. Fournet. Mobile Values, New Names and Secure Communication. POPL'01, Proceedings, 104-115.
2. M. Abadi, A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. Information and Computation, 148(1):1-70, Academic Press, 1999.
3. M. Abadi, A.D. Gordon. A Bisimulation Method for Cryptographic Protocols. Nordic Journal of Computing, 5(4):267-303, 1998.
4. M. Boreale. Symbolic trace analysis of cryptographic protocols. ICALP'01, LNCS 2076, pp.667-681, Springer-Verlag, 2001.
5. M. Boreale, R. De Nicola. A Symbolic Semantics for the π-calculus. Information and Computation, vol.126, pp.34-52, 1996.
6. M. Boreale, R. De Nicola, R. Pugliese. Proof Techniques for Cryptographic Processes. LICS'99, Proceedings, IEEE Computer Society Press, pp.157-166, 1999. Full version to appear in SIAM Journal on Computing.
7. M. Boreale, R. De Nicola, R. Pugliese. Process Algebraic Analysis of Cryptographic Protocols. Proc. of 13th FORTE / 20th PSV, Kluiver, 2000.
8. A.S. Elkjaer, M. Höhle, H. Hüttel, K.O. Nielsen. Towards Automatic Bisimilarity Checking in the Spi Calculus, *Proc. of DMTCS'99+CATS'99*, 1999.
9. M. Hennessy, H. Lin. Symbolic Bisimulations. Theoretical Computers Science, 138, pp. 353-389, 1995.
10. R. Milner. Communication and Concurrency. Prentice Hall International, 1989.
11. R. Milner. The poliadic π-calculus: a tutorial. Logic and Algebra of Specification, ed. F.L.Bauer, W.Bauer and H.Schwichtenberg Springer-Verlag, 1993.
12. R. Milner, D. Sangiorgi. Barbed Bisimulation. ICALP'92, Proceedings (W. Kuich, Ed.), LNCS 623, pp.685-695, Springer-Verlag, 1992.
13. J. Parrow, D. Sangiorgi. Algebraic theories for name-passing calculi. Information and Computation, 120, pp.174-197, 1995.