Proof Techniques for Cryptographic Processes

Michele Boreale
Dipartimento di Scienze dell'Informazione
Università di Roma "La Sapienza"
boreale@dsi.uniroma1.it

Rocco De Nicola
Dipartimento di Sistemi e Informatica
Università di Firenze
denicola@dsi.unifi.it

Rosario Pugliese
Dipartimento di Sistemi e Informatica
Università di Firenze
pugliese@dsi.unifi.it

Abstract

Contextual equivalences for cryptographic process calculi can be used to reason about correctness of protocols, but their definition suffers from quantification over all possible contexts. Here, we focus on two such equivalences, may-testing and barbed equivalence, and investigate tractable proof methods for them. To this aim, we develop an 'environment-sensitive' labelled transition system, where transitions are constrained by the knowledge the environment has of names and keys. On top of the new transition system, a trace equivalence and a co-inductive weak bisimulation equivalence are defined, both of which avoid quantification over contexts. Our main results are soundness of trace semantics and of weak bisimulation with respect to may-testing and barbed equivalence, respectively. This leads to more direct proof methods for equivalence checking. The use of such methods is illustrated via a few examples concerning implementation of secure channels by means of encrypted public channels. We also consider a variant of the labelled transition system that gives completeness, but is less handy to use.

1. Introduction

Recently, there has been much interest toward using formal methods for analysing cryptographic protocols. Here, we focus on a specific approach, which consists in modelling them as concurrent processes, described in some process calculus (like CSP [11, 17] or the spi-calculus [3, 4], a cryptographic version of the π-calculus [14]). As an example, consider the very simple protocol where two principals A and B share a private key k, and A wants to send B a datum d encrypted under k, through a public channel c:

Message 1 $\quad A \rightarrow B$: $\{d\}_k$ on c.

This informal notation can be expressed in spi-calculus as follows:

$$\begin{array}{rcl} A(d) & \stackrel{\text{def}}{=} & \overline{c}\{d\}_k.\mathbf{0} \\ B & \stackrel{\text{def}}{=} & c(x).F(x) \\ P(d) & \stackrel{\text{def}}{=} & (\nu\, k)(A(d) \mid B). \end{array}$$

Here, $\overline{c}\{d\}_k.$ means *output* along c of message $\{d\}_k$ and $\mathbf{0}$ stands for *termination*. $c(x).$ is *input* along c of a generic message x, and $F(x)$ is some expression describing the behaviour of B after the receipt of x. The whole protocol $P(d)$ is the *parallel composition* $A(d) \mid B$, with the *restriction* $(\nu\, k)$ indicating that the key k is only known to $A(d)$ and B.

The main advantage of this kind of description is that process calculi have a formal yet simple semantics, usually based on labelled transition systems (lts), that permits to make mathematically rigorous such notions as 'attacker' and 'secrecy'. Continuing the example above, a way of asserting that $P(d)$ maintains the secrecy of d is that of stating that $P(d)$ is *equivalent* to $P(d')$ for every other d'. An appropriate notion of equivalence is *may-testing* [9, 6, 3], whose intuition is precisely that no external context (which in the present setting can be read as 'attacker') may notice any difference when running in parallel with $P(d')$ or $P(d)$. A similar intuition is supported also by other contextual equivalences, like *barbed equivalence* [15]. While rigorous and intuitive, the actual definitions of these equivalences suffer from universal quantification over contexts (attackers), that makes equivalence checking very hard. It is then important to devise proof techniques that avoid such quantification. Results in this direction have been obtained for traditional process calculi (for example may testing is proved to coincide with trace equivalence in CCS [9]), but little has been done for cryptographic calculi.

In this paper, we consider may-testing and barbed equivalences for a variant of the spi-calculus with shared-key cryptographic primitives [3]. We develop an 'environment-sensitive' labelled transition system, whose transitions are constrained by the knowledge the environment has of names and keys. A trace-based equivalence and a purely co-inductive notion of weak bisimulation, that avoid quantification on contexts, are defined over the new lts and it is shown that they are sound for may-testing and barbed equivalence, respectively. We also consider a variation on the lts which gives completeness, but is less handy to use. A more detailed account of our work follows.

Let us consider the nature of the transitions in the (non-cryptographic) π-calculus. There are three kinds of basic transitions which correspond to: input of a message, output of a message and internal move. An output transition like $P \xrightarrow{(\nu\, b)\overline{a}\langle b\rangle} P'$ says that process P passes a *new* (fresh) name b ($(\nu\,\cdot)$ stands for 'new') to the environment along a channel a, and becomes P' in doing so. Once the environment acquires knowledge of b, the environment may use b arbitrarily, e.g. it can pass something back to the process along b. For example, the two-steps sequence $P \xrightarrow{(\nu\, b)\overline{a}\langle b\rangle} P' \xrightarrow{b\, c} P''$ (where $b\,c$ means 'input of c along b') is possible. At each stage, environment and process share the same knowledge of names. Thus, if the process is willing to perform some action, the environment will always be able to react. To determine whether two processes are, say, may-testing equivalent it is then sufficient to establish that they can perform the *same* sequences of (input/output) actions.

The correspondence between environment and process actions is lost when moving to the spi-calculus, i.e. when adding encryption and decryption primitives to the π-calculus. Indeed, two new facts must be taken into account.

(a) When the environment receives a new name encrypted under a fresh key, it does not acquire the knowledge of that name immediately. For instance, if P outputs a new name b encrypted under a fresh key k – $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P'$ – then name b is part of the knowledge of P', but not part of the knowledge of the environment. Thus, if P' is willing to input something at b (say $P' = b(c).P''$), the environment cannot satisfy P''s expectations: a sequence like $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P' \xrightarrow{b\, c} P''$ (that is possible in the traditional-style transition system) should not be considered as meaningful. For similar reasons, a sequence like $P \xrightarrow{(\nu\, b,k)\overline{a}\langle\{b\}_k\rangle} P' \xrightarrow{a'\, b} P''$, where the environment immediately sends the cleartext b back to the process, should not be considered as meaningful.

(b) Equivalent processes need not exhibit the same sequences of transitions. The process that performs the single output $(\nu\, k)\overline{a}\langle\{b\}_k\rangle$ and terminates, and the one that performs $(\nu\, k)\overline{a}\langle\{c\}_k\rangle$ and terminates, are equivalent, because the environment cannot distinguish between $\{b\}_k$ and $\{c\}_k$. However, the two messages can be distinguished if the environment gets the key k. Thus, processes $(\nu\, k)\overline{a}\{b\}_k.\,\overline{a}k.\,\mathbf{0}$ and $(\nu\, k)\overline{a}\{c\}_k.\,\overline{a}k.\,\mathbf{0}$ are not equivalent.

To cope with (a) above and recover the correspondence between environment and process actions, we introduce a labelled transition system that explicitly describes the knowledge of the environment. This is the main contribution of our paper. The states of the new lts are *configurations* $\sigma \rhd P$, where P is a process and σ is the current environment's knowledge, which we model as a mapping from a set of names (variables) to a set of messages. Intuitively, σ is a database of messages received from the processes, each indexed with a different name. Transitions represent interactions between the environment and P, and take the form

$$\sigma \rhd P \overset{\mu}{\underset{\delta}{\longmapsto}} \sigma' \rhd P'$$

where μ is the action of process P and δ is the complementary *environment action*. We have three different kinds of transitions.

1. The process performs an output and the environment an input. As a consequence, the environment's knowledge gets updated:
$$\sigma \rhd P \overset{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}{\underset{a(x)}{\longmapsto}} \sigma[{}^{M}\!/\!_{x}] \rhd P'$$
(here $\sigma[{}^{M}\!/\!_{x}]$ means update of σ with the new entry $[{}^{M}\!/\!_{x}]$, for a fresh variable x, and $\widetilde{b}$ is a set of fresh names). For the transition to take place, name a must belong to the knowledge of σ.

2. The process performs an input and the environment an output. As discussed previously, messages from the environment cannot be arbitrary, they must be built, via encryption and decryption, using only the knowledge in σ. Thus, transitions take the form:
$$\sigma \rhd P \overset{a\, M}{\underset{(\nu\,\widetilde{b})\overline{a}\langle\zeta\rangle}{\longmapsto}} \sigma[{}^{\widetilde{b}}\!/\!_{\widetilde{b}}] \rhd P'.$$
Informally, $\widetilde{b}$ is a set of new names just created by the environment, and ζ is an expression describing how M is built out of σ and $\widetilde{b}$. For example, if $\sigma(x_1) = \{c\}_k$ and $\sigma(x_2) = k$ and $M = c$ then ζ might be $\mathtt{dec}_{x_2}(x_1)$, indicating that message c results from decrypting the x_1-entry with the x_2-entry. Note that σ gets updated also in this case, to record the creation of the new names $\widetilde{b}$. Again, a must belong to the knowledge of σ.

3. The process performs an internal move and the environment does nothing:
$$\sigma \rhd P \overset{\tau}{\underset{-}{\longmapsto}} \sigma \rhd P'.$$

To cope with (b) when defining trace and bisimulation semantics (Section 3) on top of the new lts, the point of view is taken that *equivalent configurations should exhibit the same environment actions*. As an example, take σ with entries $\sigma(x) = a$, $\sigma(y) = b$ and $\sigma(z) = c$ and consider the configurations $C_1 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{b\}_k.\,\mathbf{0}$ and $C_2 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{c\}_k.\,\mathbf{0}$. These configurations are both trace and bisimulation equivalent, because they exhibit only the transitions:

$$C_1 \overset{(\nu\, k)\overline{a}\langle\{b\}_k\rangle}{\underset{a(w)}{\longmapsto}} \sigma[{}^{\{b\}_k}\!/\!_{w}] \rhd \mathbf{0}$$

and
$$C_2 \xmapsto[a(w)]{(\nu\, k)\overline{a}\langle\{c\}_k\rangle} \sigma[{}^{\{c\}_k}\!/\!_w] \rhd \mathbf{0}$$
which have the same environment action $a(w)$. On the other hand, as discussed above, $C_3 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{b\}_k.\,\overline{a}k.\,\mathbf{0}$ and $C_4 \overset{\text{def}}{=} \sigma \rhd (\nu\, k)\overline{a}\{c\}_k.\,\overline{a}k.\,\mathbf{0}$ should not be regarded as equivalent. Indeed, after two steps, C_3 reaches a state where the environment is $\sigma[{}^{\{b\}_k}\!/\!_w][{}^k\!/\!_v]$ that cannot be considered 'equivalent' to the environment reachable from C_4, i.e. $\sigma[{}^{\{c\}_k}\!/\!_w][{}^k\!/\!_v]$: the decryption of entry w, which is now possible because key k is known, yields distinct names (b and c). A crucial point of our approach is explaining environment equivalence by relating it to logical equivalence. Anyway, when defining both trace and bisimulation equivalence, we shall require that *matching transitions should take equivalent environments to equivalent environments*.

The trace and bisimulation equivalences avoid quantification over contexts and only require considering the actual transitions of the enriched lts. We shall show that the two equivalences imply their contextual counterparts. The converse implication fails, but we indicate how to recover completeness, at the cost of introducing a less manageable lts. To illustrate possible applications of trace and bisimulation semantics as proof techniques, in Section 4 we give a number of examples that deal with the problem of implementing secure, private channels using encrypted public channels, in the same vein of [2]. We claim that some of the equalities we establish would be very hard to prove by relying on the original, contextual definitions. Comparisons with related work are reported in Section 5.

2. The Language

Syntax (Table 1) In the standard π-calculus, names are the only transmissible objects. Here, the possibility has been added to communicate *messages* obtained via shared-key encryption: message $\{M\}_k$ represents the ciphertext obtained by encrypting message M under key k using a shared-key encryption system. *Expressions* are obtained applying encryption and decryption operators to names and ciphertexts. For example, expression $\mathtt{dec}_{\zeta_2}(\zeta_1)$ represents the text obtained by decrypting the ciphertext ζ_1 under the key ζ_2. Logical *formulae* generalize the usual matching operator of the π-calculus with a predicate $name(\cdot)$, which tests whether the argument is a plain name or a compound ciphertext, and with a 'let' construct that binds the value of some expression ζ to a name z. *Processes* are built using a set of operators which include those from the standard π-calculus, plus two new operators: boolean guard and encryption/decryption.

There are a few differences from Abadi and Gordon's spi-calculus [3]. First, our encryption/decryption operator has the form $\mathtt{let}\; z = \zeta \;\mathtt{in}\, P$: it attempts evaluating ζ; if the evaluation succeeds, the value of ζ is bound to z within P, otherwise the whole process is stuck. This enables one to write process expressions in a more compact form: for instance, the spi-calculus process $\mathtt{case}\; M \;\mathtt{of}\; \{k\}_h \;\mathtt{in}\; (\mathtt{case}\; N \;\mathtt{of}\; \{z\}_k \;\mathtt{in}\; P)$ can be written as $\mathtt{let}\; z = \mathtt{dec}_{\mathtt{dec}_h(M)}(N) \;\mathtt{in}\; P$ in our syntax. A more relevant difference from the spi-calculus is that in our syntax tuples are ruled out and that decryption keys cannot be compound messages: these simplifications permit a clearer presentation of our approach. Finally, we have included non-deterministic choice ($+$), which is sometimes useful for specification purposes.

The notions of *free names* of a process P, $fn(P)$, of *bound names* of P, $bn(P)$, and of α-equivalence arise as expected; $n(P)$ is $fn(P) \cup bn(P)$. We shall write $fn(P,Q)$ in place of $fn(P) \cup fn(Q)$ (similarly for $bn(\cdot)$ and $n(\cdot)$). We assume the usual conventions on α-equivalence and bound names; in particular, we identify α-equivalent terms and assume that bound names are always fresh and distinct.

A *substitution* σ is a finite partial map from $\mathcal{N}$ to $\mathcal{M}$; the domain and proper co-domain of σ are written $dom(\sigma)$ and $range(\sigma)$. The substitution mapping x to M is written $[{}^M\!/\!_x]$, while the substitution mapping x_i to M_i, for $i \in I$, is written as $[{}^{M_i}\!/\!_{x_i}]_{i \in I}$. For x not appearing in σ, we write $\sigma[{}^M\!/\!_x]$ for the substitution which is the union of σ and $[{}^M\!/\!_x]$. For a given $V \subseteq_{\text{fin}} \mathcal{N}$, we write ϵ_V for the substitution with $dom(\epsilon_V) = V$ that acts as the identity on V. For any term t, $t\sigma$ denotes the result of simultaneously replacing each $x \in fn(t) \cap dom(\sigma)$ with $\sigma(x)$, with renaming of bound names of t possibly involved to avoid captures.

We shall often abbreviate $\alpha.\mathbf{0}$ as α, where α is an input or output prefix, and $(\nu\, a)(\nu\, b)P$ as $(\nu\, a, b)P$. We shall use the tilde $\widetilde{\cdot}$ to denote tuples of objects; e.g. $\widetilde{x}$ is a generic tuple of names. All our notations are extended to tuples component-wise. If $\widetilde{k} = (k_1, \ldots, k_n)$, then $\{M\}_{\widetilde{k}}$ stands for $\{\cdots\{M\}_{k_1}\cdots\}_{k_n}$ (similarly for $\mathtt{dec}_{\widetilde{k}}(M)$).

Standard operational semantics Consider the closure of $\mathcal{P}$ under the set of all substitutions, that is the set of terms $\mathcal{P}^c \overset{\text{def}}{=} \{P\sigma : P \in \mathcal{P}$ and σ is a substitution$\}$ (similarly for Φ^c). While we are primarily interested in $\mathcal{P}$, for technical reasons that will be apparent later, it is convenient to define the operational and observational semantics over the larger $\mathcal{P}^c$. Therefore, in the rest of the paper, a 'process' is a term of $\mathcal{P}^c$ and $P, Q, \ldots$ range over $\mathcal{P}^c$. In order to define the (traditional) operational semantics of the calculus, we need two evaluation functions: one for expressions, the other for formulae. The *evaluation function* for expressions, $\widehat{\cdot} : \mathcal{Z} \to \mathcal{M} \cup \{-\}$ (where $-$ is a distinct symbol), is defined by induction on ζ as follows:

- $\widehat{a} = a$
- $\widehat{\{\zeta_1\}_{\zeta_2}} = \begin{cases} \{M\}_k & \text{if } \widehat{\zeta_1} = M \text{ and } \widehat{\zeta_2} = k \in \mathcal{N}, \\ & \text{for some } M \text{ and } k \\ - & \text{otherwise} \end{cases}$

$a, b \ldots, h, k, \ldots, x, y, z \ldots$			*names* $\mathcal{N}$
M	$::=$	$a \mid \{M\}_k$	*messages* $\mathcal{M}$
ζ	$::=$	$a \mid \{\zeta_1\}_{\zeta_2} \mid \mathtt{dec}_{\zeta_2}(\zeta_1)$	*expressions* $\mathcal{Z}$
ϕ	$::=$	$tt \mid name(M) \mid [M_1 = M_2]$	*formulae* Φ
		$\mid \mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi \mid \phi_1 \wedge \phi_2 \mid \neg\phi$	
P	$::=$		*processes* $\mathcal{P}$
		$\mathbf{0}$	$(null)$
	$\mid$	$a(x).P$	$(input\ prefix)$
	$\mid$	$\overline{a}M.P$	$(output\ prefix)$
	$\mid$	$P_1 + P_2$	$(non-deterministic\ choice)$
	$\mid$	$P_1 \mid P_2$	$(parallel\ composition)$
	$\mid$	$(\nu\, a)P$	$(restriction)$
	$\mid$	$!P$	$(replication)$
	$\mid$	ϕP	$(boolean\ guard)$
	$\mid$	$\mathtt{let}\ z = \zeta\ \mathtt{in}\ P$	$(encryption/decryption)$

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 $\mathtt{let}\ z = \zeta\ \mathtt{in}\ \cdot$, we assume that $z \notin n(\zeta)$.

Table 1. Syntax of the calculus

- $$\widehat{\mathtt{dec}_{\zeta_2}(\zeta_1)} = \begin{cases} M & \text{if } \widehat{\zeta_1} = \{M\}_k \text{ and } \widehat{\zeta_2} = k \in \mathcal{N}, \\ & \text{for some } M \text{ and } k \\ - & \text{otherwise.} \end{cases}$$

The evaluation of an expression 'fails', i.e. returns the value $-$, whenever an encryption/decryption with something different from a name is attempted, or whenever a decryption with something different from the encryption key is attempted. For instance, the evaluation of $\{a\}_{\{b\}_c}$ and of $\mathtt{dec}_b(\{a\}_c)$ is $-$, while $\mathtt{dec}_c(\{a\}_c)$ evaluates to a.

The evaluation function for formulae, $[\![\cdot]\!] : \Phi^c \rightarrow \{tt, ff\}$, is defined by induction on ϕ as expected (evaluation of $[\![\ \mathtt{let}\ z = \zeta\ \mathtt{in}\ \phi\ ]\!]$ yields ff if $\widehat{\zeta} = -$, $[\![\ \phi[\widehat{\zeta}/z]\ ]\!]$ otherwise). For any substitution σ, we let $\sigma \models \phi$ mean that $[\![\ \phi\sigma\ ]\!] = tt$.

The (traditional) *operational semantics* (Table 2) is defined by the familiar early-style inference rules of the π-calculus, plus a rule for encryption/decryption (this presentation is essentially equivalent to the 'commitment relation' semantics of Abadi and Gordon in [3]). Process *actions* (i.e. labels of the transition system), ranged over by $\mu, \lambda, \ldots$, can be of three forms: τ (internal action), $a\, M$ (input at a where message M is received) and $(\nu\,\widetilde{b})\overline{a}\langle M\rangle$ (output at a where message M containing the bound names $\widetilde{b}$ is sent). We write $\overline{a}\langle M\rangle$ instead of $(\nu\,\widetilde{b})\overline{a}\langle M\rangle$ whenever $\widetilde{b} = \emptyset$. Input and output actions will be called *visible* actions. We use s to range over sequences of visible actions (traces), and write $\Rightarrow$ or $\overset{\epsilon}{\Longrightarrow}$ to denote the reflexive and transitive closure of $\xrightarrow{\tau}$ and, inductively, $\overset{s}{\Longrightarrow}$ for $\Rightarrow\xrightarrow{\mu}\overset{s'}{\Longrightarrow}$ when $s = \mu \cdot s'$. $P \overset{s}{\Longrightarrow}$ means that $P \overset{s}{\Longrightarrow} P'$ for some P'.

Remark 2.1 Note that some process terms in $\mathcal{P}$ may reduce to terms in $\mathcal{P}^c$, like in $\overline{a}\{b\}_k.P \mid a(x).\overline{x}c \xrightarrow{\tau} P \mid \overline{\{b\}_k}c$. This explains why it is convenient to consider $\mathcal{P}^c$ in place of $\mathcal{P}$. Also note that, similarly to [3], a term like $\overline{\{b\}_k}c$, which tries to use a compound message as a channel, is stuck. Differently from [3], a term like $\overline{a}\{b\}_{\{c\}_k}$ is stuck in our operational semantics, as $\{b\}_{\{c\}_k}$ is not a message (a compound $\{c\}_k$ occurs in place of a plain name).

May-testing and barbed equivalence We now instantiate the general framework of may-testing [9] to our calculus. *Observers* are processes that can perform a distinct 'success' action ω, to signal that some test has been successfully passed by the observed process. For instance, the observer $(\nu\, b)\overline{a}b.\, b(x).\, [x = c]\omega$ tests for the ability of a process to receive a new name b on channel a and then to send name c along this b. The may-testing preorder can be defined in terms of the ability of processes to pass tests proposed by observers.

Definition 2.2 (may-testing preorder, [9])
We write $P \underset{\sim}{\sqsubset} Q$ to mean that, for every observer O, if $P \mid O \overset{\omega}{\Longrightarrow}$ then $Q \mid O \overset{\omega}{\Longrightarrow}$. $\diamond$

The equivalence obtained as the kernel of the preorder $\underset{\sim}{\sqsubset}$ is denoted by $\simeq$ ($\simeq\ =\ \underset{\sim}{\sqsubset} \cap \underset{\sim}{\sqsubset}^{-1}$).

We switch now to barbed equivalence [15]. The intuition is somehow similar to that of testing, but in addition barbed equivalence conveys a notion of fairness, which derives from the co-inductive part of its definition. In what

(INP)	$a(x).\,P \xrightarrow{a\,M} P[^M/_x]$	(OUT)	$\overline{a}M.\,P \xrightarrow{\overline{a}\langle M\rangle} P$
(SUM)	$\dfrac{P_1 \xrightarrow{\mu} P_1'}{P_1 + P_2 \xrightarrow{\mu} P_1'}$	(REP)	$\dfrac{P \mid\, !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'}$
(PAR)	$\dfrac{P_1 \xrightarrow{\mu} P_1'}{P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2}$	(COM)	$\dfrac{P_1 \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P_1' \qquad P_2 \xrightarrow{a\,M} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} (\nu\,\widetilde{b})(P_1' \mid P_2')}$
(RES)	$\dfrac{P \xrightarrow{\mu} P'}{(\nu\,c)P \xrightarrow{\mu} (\nu\,c)P'}\, c \notin n(\mu)$	(OPEN)	$\dfrac{P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P'}{(\nu\,c)P \xrightarrow{(\nu\,\widetilde{b}c)\overline{a}\langle M\rangle} P'}\, c \neq a, c \in n(M) - \widetilde{b}$
(GUARD)	$\dfrac{[\![\,\phi\,]\!] = tt \qquad P \xrightarrow{\mu} P'}{\phi P \xrightarrow{\mu} P'}$	(LET)	$\dfrac{\widehat{\zeta} \neq - \qquad P[\widehat{\zeta}/z] \xrightarrow{\mu} P'}{\texttt{let}\ z = \zeta\ \texttt{in}\ P \xrightarrow{\mu} P'}$

Table 2. Standard operational semantics (symmetric versions of (SUM), (PAR) and (COM) omitted)

follows, we say that a process P *commits to* a, and write $P \downarrow a$, if $P \xrightarrow{a\,M}$ or $P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle}$, for some M and $\widetilde{b}$. We also write $P \Downarrow a$ if $P \Longrightarrow P'$ and $P' \downarrow a$, for some P'.

Definition 2.3 (barbed equivalence, [15])
A symmetric relation of processes $\mathcal{S}$ is a *barbed bisimulation* if whenever $P\mathcal{S}Q$ then: (1) whenever $P \xrightarrow{\tau} P'$ then there is Q' such that $Q \Longrightarrow Q'$ and $P'\mathcal{S}Q'$, and (2) whenever $P \downarrow a$ then $Q \Downarrow a$. *Barbed bisimilarity*, written $\dot{\simeq}$, is the largest barbed bisimulation. Two processes P and Q are *barbed equivalent*, written $P \cong Q$, if for all R we have that $P \mid R \dot{\simeq} Q \mid R$. $\diamond$

It is worthwhile to notice that neither $\underset{\sim}{\sqsubset}$ nor $\cong$ are (pre-)congruences, due to the usual problems arising with input prefix (see [14]).

3. Trace and Bisimulation Semantics

We introduce now the new 'environment-sensitive' lts. Then, we will define trace and bisimulation semantics, and show that they are sound for may-testing and barbed equivalence, respectively. Finally, we consider a more complex lts which guarantees completeness.

An environment-sensitive lts We start by making precise the concept of knowledge of an environment σ, that is, all the information that can be deduced from the content of σ.

Definition 3.1 (environment knowledge)
Consider $W \subseteq \mathcal{M}$. The set $Kn(W)$ is defined as the smallest $S \subseteq \mathcal{M}$ such that: (i) $W \subseteq S$, and (ii) whenever $k \in S$ and $\{M\}_k \in S$ then $M \in S$. We define $kn(W) \stackrel{\text{def}}{=} Kn(W) \cap \mathcal{N}$. Given a substitution σ, $Kn(\sigma) \stackrel{\text{def}}{=} Kn(range(\sigma))$ and $kn(\sigma) \stackrel{\text{def}}{=} kn(range(\sigma))$. $\diamond$

For instance, $kn([^{\{b\}_{(k,l,h)}}/_w, ^{\{a\}_k}/_x, ^{\{k\}_h}/_y, ^h/_z]) = \{a, h, k\}$. Note that $kn(\sigma)$ can be easily computed in a finite number of steps. We are now ready to introduce the new lts. States are *configurations* of the form $\sigma \rhd P$, where σ represents the environment, and transitions take the form $\sigma \rhd P \underset{\delta}{\overset{\mu}{\longmapsto}} \sigma' \rhd P'$ and represent atomic interactions between the process and the environment. The environment action δ can be of three forms, output, input and 'no action':

$$\delta ::= (\nu\,\widetilde{b})\overline{a}\langle\zeta\rangle \quad|\quad a(x) \quad|\quad - \,.$$

Free names and bound names of δ are defined as expected, in particular $bn(a(x)) = \{x\}$. The inference rules for the transition relation $\underset{\delta}{\overset{\mu}{\longmapsto}}$ are displayed in Table 3. The *visible* environment actions are input and output; u ranges over sequences of visible environment actions. In the following, we write $\Longmapsto$ or $\overset{\epsilon}{\Longmapsto}$ to denote the reflexive and transitive closure of $\underset{-}{\overset{\tau}{\longmapsto}}$ and, inductively, write $\underset{u}{\overset{s}{\Longmapsto}}$ for $\Longmapsto\underset{\delta}{\overset{\mu}{\longmapsto}}\underset{u'}{\overset{s'}{\Longmapsto}}$ if $s = \mu \cdot s'$ and $u = \delta \cdot u'$.

Trace and bisimulation semantics In order to define observational semantics based on $\underset{u}{\overset{s}{\Longmapsto}}$, we have to precisely define when two environments, represented by substitutions σ and σ', are 'equivalent'. Informally, σ and σ' are equivalent whenever they are logically indistinguishable, i.e. whenever for each formula ϕ with $fn(\phi) \subseteq dom(\sigma)$, $\sigma \models \phi$ if and only if $\sigma' \models \phi$. This logical characterization is difficult to check, as it contains a quantification on all formulae. Below, we give a definition, that is easy to check and still implies logical indistinguishability. First, a notational shorthand. Given a $\sigma = [^{M_i}/_{x_i}]_{i\in I}$ and $i \in I$, we denote by $core(\sigma, x_i)$ whatever cannot be further decrypted in M_i using the knowledge available in σ. Formally, we

$$\text{(E-OUT)}\ \frac{P \xrightarrow{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} P' \qquad a \in kn(\sigma)}{\sigma \rhd P \xmapsto[a(x)]{(\nu\,\widetilde{b})\overline{a}\langle M\rangle} \sigma[{}^{M}/_{x}] \rhd P'} \qquad\qquad \text{(E-TAU)}\ \frac{P \xrightarrow{\tau} P'}{\sigma \rhd P \xmapsto[-]{\tau} \sigma \rhd P'}$$

$$\text{(E-INP)}\ \frac{P \xrightarrow{a\ M} P' \quad a \in kn(\sigma) \quad M = \widehat{\zeta\sigma} \quad \widetilde{b} \stackrel{\text{def}}{=} (n(\zeta) - dom(\sigma))}{\sigma \rhd P \xmapsto[(\nu\,\widetilde{b})\overline{a}\langle\zeta\rangle]{a\ M} \sigma[{}^{\widetilde{b}}/_{\widetilde{b}}] \rhd P'}$$

Table 3. Rules for the environment-sensitive lts

define $core(\sigma, x_i)$ as the message N such that, for some $\widetilde{k} \subseteq kn(\sigma)$, it holds $M_i = \{N\}_{\widetilde{k}}$ and either N is a name, or $N = \{N'\}_h$ for some N' and $h \notin kn(\sigma)$. For example, given $\sigma = [{}^{\{a\}_{(h,k)}}/_{x_1},\ {}^{k}/_{x_2}]$, then $core(\sigma, x_1) = \{a\}_h$. The rationale of the definition below is that it should not be possible to distinguish entries of two equivalent environments by failure of decryption (a), nor by the $name(\cdot)$ predicate (b), nor by the equality predicate (c).

Definition 3.2 (equivalent environments) Let $\sigma = [{}^{M_i}/_{x_i}]_{i\in I}$ and $\sigma' = [{}^{M'_i}/_{x_i}]_{i\in I}$ be two substitutions with the same domain. For each $i \in I$, let $N_i = core(\sigma, x_i)$ and $N'_i = core(\sigma', x_i)$. We say that σ and σ' are *equivalent*, and write $\sigma \sim \sigma'$, if for each $i \in I$ the following conditions hold:

(a) for some $\widetilde{k}_i$, $M_i = \{N_i\}_{\widetilde{k}_i}$ and $M'_i = \{N'_i\}_{\widetilde{k}_i}$;

(b) if $N_i \in \mathcal{N}$ or $N'_i \in \mathcal{N}$ then $N_i = N'_i$;

(c) for each $j \in I$, $N_i = N_j$ iff $N'_i = N'_j$. $\diamond$

As an example, $\sigma_1 = [{}^{b}/_{x_1},\ {}^{c}/_{x_2},\ {}^{\{b\}_k}/_{x_3}]$ and $\sigma_2 = [{}^{b}/_{x_1},\ {}^{c}/_{x_2},\ {}^{\{c\}_k}/_{x_3}]$ are equivalent. On the contrary, $\sigma_3 = \sigma_1[{}^{k}/_{x_4}]$ and $\sigma_4 = \sigma_3[{}^{k}/_{x_4}]$ are not equivalent, because $core(\sigma_3, x_3) = b \neq c = core(\sigma_4, x_3)$, thus condition (b) is violated. It is easy to prove that if $\sigma \sim \sigma'$ then $kn(\sigma) = kn(\sigma')$. We are now ready to define a trace-based preorder.

Definition 3.3 (trace preorder)
Let $\sigma_1 \sim \sigma_2$. Given two processes P and Q, we write $(\sigma_1, \sigma_2) \vdash P \ll Q$ if whenever $\sigma_1 \rhd P \overset{s}{\underset{u}{\Longmapsto}} \sigma'_1 \rhd P'$ then[1] there are s', σ'_2 and Q' such that $\sigma_2 \rhd Q \overset{s'}{\underset{u}{\Longmapsto}} \sigma'_2 \rhd Q'$ and $\sigma'_1 \sim \sigma'_2$. $\diamond$

Note that we just require that the environment trace (u) of matching transitions is the same, and do not require anything about the process traces (s and s'). As an example, define $\sigma = [{}^{a}/_{x},\ {}^{b}/_{y},\ {}^{c}/_{z}]$. Then $\sigma \rhd (\nu\,k)\overline{a}\{b\}_k$ and $\sigma \rhd (\nu\,k)\overline{a}\{c\}_k$ are $\ll$-equivalent. On the contrary, $\sigma \rhd (\nu\,k)\overline{a}\{b\}_k.\overline{a}k$ and $\sigma \rhd (\nu\,k)\overline{a}\{c\}_k.\overline{a}k$ are not related, because $\sigma[{}^{\{b\}_k}/_{v},\ {}^{k}/_{w}] \not\sim \sigma[{}^{\{c\}_k}/_{v},\ {}^{k}/_{w}]$, for any v, w. More examples will be given in Section 4.

Bisimulation over the environment-sensitive lts can be very easily defined. In what follows $\sigma \rhd P \overset{\widehat{\mu}}{\underset{\delta}{\Longmapsto}} \sigma' \rhd P'$ stands for $\sigma \rhd P \overset{\mu}{\underset{\delta}{\Longmapsto}} \sigma' \rhd P'$ if $\mu \neq \tau$, and for $\sigma \rhd P \Longmapsto \sigma' \rhd P'$ if $\mu = \tau$. We say that a pair of configurations $(\sigma_1 \rhd P, \sigma_2 \rhd Q)$ is *compatible* if σ_1 and σ_2 are equivalent. We write $(\sigma_1, \sigma_2) \vdash P\,\mathcal{R}\,Q$ if $(\sigma_1 \rhd P)\,\mathcal{R}\,(\sigma_2 \rhd Q)$, for a binary relation $\mathcal{R}$.

Definition 3.4 (weak bisimulation) Let $\mathcal{R}$ be a relation of compatible pairs of configurations. We say that $\mathcal{R}$ is a *weak bisimulation* if whenever $(\sigma_1, \sigma_2) \vdash P\,\mathcal{R}\,Q$ and $\sigma_1 \rhd P \xmapsto[\delta]{\mu} \sigma'_1 \rhd P'$ then there are μ', σ'_2 and Q' such that $\sigma_2 \rhd Q \overset{\widehat{\mu'}}{\underset{\delta}{\Longmapsto}} \sigma'_2 \rhd Q'$ and $(\sigma'_1, \sigma'_2) \vdash P'\,\mathcal{R}\,Q'$, and the converse on the transitions of Q and P. *Bisimilarity*, $\approx$, is the largest weak bisimulation relation. $\diamond$

Soundness It is convenient to state the soundness theorems for notions which are more general than $\sqsubseteq\!\!\!\!\sim$ and $\cong$. For equivalent σ_1 and σ_2, we let $(\sigma_1, \sigma_2) \vdash P \sqsubseteq\!\!\!\!\sim Q$ mean that for each observer O with $fn(O) \subseteq dom(\sigma_1)$, if $P|O\sigma_1 \stackrel{\omega}{\Longrightarrow}$ then $Q \mid O\sigma_2 \stackrel{\omega}{\Longrightarrow}$. Clearly, $P \sqsubseteq\!\!\!\!\sim Q$ holds if and only if $(\epsilon_V, \epsilon_V) \vdash P \sqsubseteq\!\!\!\!\sim Q$ for some $V \supseteq fn(P, Q)$. A similar generalization can also be given for barbed equivalence (just close under contexts $R\sigma_1$ and $R\sigma_2$ with $fn(R) \subseteq dom(\sigma_1)$ and check only for those commitments $\downarrow a$ such that a is in the knowledge of σ_1 and σ_2): due to lack of space we omit the details.

Theorem 3.5 (soundness of trace equivalence)
If $(\sigma_1, \sigma_2) \vdash P \ll Q$ then $(\sigma_1, \sigma_2) \vdash P \sqsubseteq\!\!\!\!\sim Q$.

PROOF: Assume $fn(O) \subseteq dom(\sigma_1)$ and $P \mid O\sigma_1 \stackrel{\omega}{\Longrightarrow}$. The latter sequence can be 'unzipped' as: $P \stackrel{s}{\Longrightarrow} P'$, $O\sigma_1 \stackrel{r\omega}{\Longrightarrow}$ with s and r complementary. Under this condition, it can be deduced that $\sigma_1 \rhd P \overset{s}{\underset{u}{\Longmapsto}} \sigma'_1 \rhd P'$, for σ'_1 and u s.t. $r = \widehat{u\sigma'_1}$ (here $\widehat{\cdot}$ denotes the obvious extension

[1] Recall that bound names of s and u are assumed to be fresh. A similar remark applies to μ and δ in Def. 3.4.

of the expression evaluation function). Then by hypothesis $\sigma_2 \rhd Q \overset{s'}{\underset{u}{\Longmapsto}} \sigma'_2 \rhd Q'$, with $\sigma'_1 \sim \sigma'_2$, and moreover s' is complementary to $r' \stackrel{\text{def}}{=} \widehat{u\sigma'_2}$. The crucial point is now that $O\sigma_1 \xRightarrow{r\omega}$ implies $O\sigma_2 \xRightarrow{r'\omega}$ (this property relies on the logical indistinguishability of σ'_1 and σ'_2). The thesis follows from this and from $Q \xRightarrow{s'} Q'$. $\square$

The purely co-inductive proof technique of bisimulation can be enhanced tailoring to the present setting the so called *up-to* techniques (similar to those in, e.g., [16, 7]), which often permit reducing the size of the relation to exhibit. For example, the 'up to parallel composition' technique permits cutting away common contexts from process derivatives: a relation $\mathcal{R}$ is a *weak bisimulation up to parallel composition* if $\mathcal{R}$ satisfies the definition of weak bisimulation (Def. 3.4), but with the condition on the derivatives '$(\sigma'_1, \sigma'_2) \vdash P' \,\mathcal{R}\, Q'$' replaced by the (weaker) '$(\sigma'_1, \sigma'_2) \vdash P' \,\mathcal{R}\, Q'$–up to parallel composition'. The latter means that there are P_0, Q_0 and R with $fn(R) \subseteq dom(\sigma'_1)$ such that: $P' = P_0 \,|\, R\sigma'_1$ and $Q' = Q_0 \mid R\sigma'_2$ and $(\sigma'_1, \sigma'_2) \vdash P_0 \,\mathcal{R}\, Q_0$.

Another technique, 'up to contraction', permits to discard environments entries that give redundant information, while a third one, 'up to structural congruence', permits freely identifying processes up to structural congruence $\equiv$ [13]. Due to lack of space, we omit formal definitions of these techniques. It can be proven that each 'bisimulation up-to' defined above, and any combination of them[2], is included in $\approx$. We shall see an example of use of these technique in Section 4.

Theorem 3.6 (soundness of weak bisimilarity)
If $(\sigma_1, \sigma_2) \vdash P \approx Q$ then $(\sigma_1, \sigma_2) \vdash P \cong Q$.

PROOF: Consider the least relation $\mathcal{R}$ such that: if $(\sigma_1, \sigma_2) \vdash P \approx Q$ and $fn(R) \subseteq dom(\sigma_1)$ then $(\sigma_1, \sigma_2) \vdash (P \mid R\sigma_1) \,\mathcal{R}\, (Q \mid R\sigma_2)$. Show that $\mathcal{R} \subseteq \approx$. The proof relies heavily on the up-to techniques mentioned above, or variations on them. The thesis follows because $\approx \subseteq \dot{\cong}$. $\square$

Completeness To see that $\ll$ does not coincide with $\underset{\approx}{\sqsubset}$, consider $\sigma = [a/x]$; then taken a new name w, $(\sigma, \sigma) \vdash \overline{a}w \simeq (\nu\, w)\overline{a}w$, but the two processes are easily seen to be *not* related by $\ll$ (the same example holds for $\approx$). The reason is that the definition of $\sigma_1 \sim \sigma_2$ is too demanding on the identity of known names, in that $kn(\sigma_1) = kn(\sigma_2)$. We can relax this condition by modifying Definition 3.2 as we do below. We use the following notation: given two tuples $\widetilde{x} = x_{i \in I}$ and $\widetilde{J} = (j_1, \ldots, j_k) \subseteq I$, we let $\widetilde{x}[\widetilde{J}]$ denote the tuple $(x_{j_1}, \ldots, x_{j_k})$. For instance, if $\widetilde{x} = (x_1, x_2, x_3)$ and $\widetilde{J} = (1, 1, 3)$ then $\widetilde{x}[\widetilde{J}] = (x_1, x_1, x_3)$.

[2]Formally, each up-to technique can be described as a functional over binary relations, and a combination of techniques is simply a composition of the corresponding functionals; see [16, 7].

Definition 3.7 (equivalent environments revised)
Let $\sigma = [M_i/x_i]_{i \in I}$ and $\sigma' = [M'_i/x_i]_{i \in I}$ be two substitutions with the same domain. For each $i \in I$, let $N_i = core(\sigma, x_i)$ and $N'_i = core(\sigma', x_i)$, and let $\widetilde{N} = N_{i \in I}$ and $\widetilde{N'} = N'_{i \in I}$. We say that σ and σ' are *equivalent*, and write $\sigma \sim \sigma'$, if for each $i \in I$ the following conditions hold:

(a) for some tuple $\widetilde{J}_i \subseteq I$, $M_i = \{N_i\}_{\widetilde{N}[\widetilde{J}_i]}$ and $M'_i = \{N'_i\}_{\widetilde{N'}[\widetilde{J}_i]}$;

(b) $N_i \in \mathcal{N}$ iff $N'_i \in \mathcal{N}$;

(c) for each $j \in I$, $N_i = N_j$ iff $N'_i = N'_j$. $\diamond$

As an example, the environments $[\{b\}_{(k,h)}/x_1,\ h/x_2]$ and $[\{b\}_{(k',h')}/x_1,\ h'/x_2]$ are equivalent, according to the new definition. Also the definition of $\sigma \rhd P \overset{\mu}{\underset{\delta}{\longmapsto}} \sigma' \rhd P'$ (Table 3) requires an adjustment: we relax the condition that δ has the same subject name as μ (that is a) and just require that the subject of δ is an expression η s.t. $n(\eta) \subseteq dom(\sigma)$ and $\widehat{\eta\sigma} = a$. Based on these two new notions, the definitions of the new trace preorder, $\ll'$, and of the new bisimulation equivalence, $\approx'$, remain formally unchanged. It is not difficult to prove that $\ll'$ coincides with $\underset{\approx}{\sqsubset}$, and, for image-finite processes (see [12]), that $\approx'$ coincides with $\cong$. We omit the details and state:

Theorem 3.8 Let $\sigma_1 \sim \sigma_2$. Then $(\sigma_1, \sigma_2) \vdash P \ll' Q$ iff $(\sigma_1, \sigma_2) \vdash P \underset{\approx}{\sqsubset} Q$. For image-finite processes P and Q, $(\sigma_1, \sigma_2) \vdash P \approx' Q$ iff $(\sigma_1, \sigma_2) \vdash P \cong Q$.

For the rest of the paper, we shall stick to the more manageable relations $\ll$ and $\approx$.

4. Applications

The congruence laws listed in Table 4 are very useful (especially (C-PAR) and (C-RES)) because they permit a kind of compositional reasoning, as we shall see in the examples of this section. They are stated for $\ll$, but are valid also for $\approx$. Another useful fact is that structural congruence $\equiv$ [13] is included in both $\approx$ and $\ll$.

In the following examples, we show a possible use of our framework for proving security properties of communication protocols. In the same vein of [1, 2], the idea is that of implementing communication on secure (private) channels by means of encrypted communication on public channels. Let us consider the π-calculus process:

$$P \stackrel{\text{def}}{=} (\nu\, c)(\overline{c}d \mid c(z).\, R)$$

where c does not occur in R. Process P creates a private channel c which is used to transmit name d. Communication on c is secure because the execution context does not know the private channel c. Since P consists of two concurrent subprocesses, the actual implementation could allocate them onto two different computers, whose interconnections

(C-INP) Suppose that for all ζ such that $\widetilde{y} \overset{\text{def}}{=} (n(\zeta) - dom(\sigma_1))$ are fresh and $\widehat{\zeta\sigma_1} \neq -$ it holds: $(\sigma_1[\widetilde{y}/\widetilde{y}], \sigma_2[\widetilde{y}/\widetilde{y}]) \vdash P[\widehat{\zeta\sigma_1}/x] \ll Q[\widehat{\zeta\sigma_2}/x]$. Then $(\sigma_1, \sigma_2) \vdash a(x).P \ll a(x).Q$.

(C-OUT) If $(\sigma_1[M_1/x], \sigma_2[M_2/x]) \vdash P \ll Q$ then $(\sigma_1[M_1/x], \sigma_2[M_2/x]) \vdash \overline{a}M_1.P \ll \overline{a}M_2.Q$.

(C-PAR) Suppose that $fn(R) \subseteq dom(\sigma_1)$.
If $(\sigma_1, \sigma_2) \vdash P \ll Q$ then $(\sigma_1, \sigma_2) \vdash P \mid R\sigma_1 \ll Q \mid R\sigma_2$.

(C-RES) Suppose that $(\sigma_1[M_1/x], \sigma_2[M_2/x]) \vdash P \ll Q$.
If $\widetilde{k} \cap n(\sigma_1) = \emptyset$ and $\widetilde{h} \cap n(\sigma_2) = \emptyset$ then $(\sigma_1, \sigma_2) \vdash (\nu\, \widetilde{k})P \ll (\nu\, \widetilde{h})Q$.

Table 4. Some congruence rules

are not guaranteed to be secure. Communication on c has to be implemented in terms of lower-level, encrypted communication on some public channel, say p. Thus, process P might be implemented as

$$I_P \overset{\text{def}}{=} (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R).$$

In I_P, name k_c is a private encryption key that corresponds to channel c. Note that this implementation does not guarantee that d will eventually be passed to R: message $\{d\}_{k_c}$ could be captured by some context (attacker) listening at p. An implementation that solves this problem will be presented later.

Example 4.1 (secrecy)
Assume that R keeps z secret, i.e. for every d and d', $R[d/z]$ is may-equivalent to $R[d'/z]$. Under this hypothesis, we want to prove that the implementation scheme for P preserves secrecy. To see this, we consider a generic d', let $Q \overset{\text{def}}{=} (\nu\, c)(\overline{c}d' \mid c(z).\, R)$ and show that:

$$(\epsilon_V, \epsilon_V) \vdash I_P \simeq I_Q$$

where I_Q is the obvious implementation of Q and $V = fn(I_P, I_Q)$. In order to prove this, let y be any fresh name and define $\sigma_1 \overset{\text{def}}{=} \epsilon_V[\{d\}_{k_c}/y]$ and $\sigma_2 \overset{\text{def}}{=} \epsilon_V[\{d'\}_{k_c}/y]$. First, rule (C-INP) allows one to prove that $(\sigma_1, \sigma_1) \vdash p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R \simeq p(x).[x = \{d\}_{k_c}]R[d/z]$ (to prove this, one exploits the fact that for any ζ s.t. $n(\zeta) - dom(\sigma_1)$ are fresh, if $\widehat{\zeta\sigma_1} = \{M\}_{k_c}$ then $M = d$). This fact and (C-PAR) are used to infer that:

$$(\sigma_1, \sigma_1) \vdash \overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R \simeq \overline{p}\{d\}_{k_c} \mid p(x).[x = \{d\}_{k_c}]R[d/z] \qquad (1)$$

Considering d' in place of d, symmetrically one proves that:

$$(\sigma_2, \sigma_2) \vdash \overline{p}\{d'\}_{k_c} \mid p(x).[x = \{d'\}_{k_c}]R[d'/z] \simeq \overline{p}\{d'\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R \quad (2)$$

Now, from $R[d/z] \simeq R[d'/z]$ and (C-INP), it follows that

$$(\sigma_1, \sigma_2) \vdash p(x).[x = \{d\}_{k_c}]R[d/z] \simeq p(x).[x = \{d'\}_{k_c}]R[d'/z],$$

from which, applying (C-PAR) one gets:

$$(\sigma_1, \sigma_2) \vdash \overline{p}\{d\}_{k_c} \mid p(x).[x = \{d\}_{k_c}]R[d/z] \simeq \overline{p}\{d'\}_{k_c} \mid p(x).[x = \{d'\}_{k_c}]R[d'/z] \qquad (3)$$

Now, from (1), (3) and transitivity, one gets that

$$(\sigma_1, \sigma_2) \vdash \overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R \simeq \overline{p}\{d'\}_{k_c} \mid p(x).[x = \{d'\}_{k_c}]R[d'/z].$$

From this fact, (2) and transitivity one gets that

$$(\sigma_1, \sigma_2) \vdash \overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R \simeq \overline{p}\{d'\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ R.$$

Finally, the wanted claim follows by applying (C-RES) (with $(\nu\, k_c)$) to the equality above. $\diamond$

Example 4.2 (may-semantics preservation)
Here we show that the previous implementation scheme also preserves may semantics. We relax the hypothesis that R keeps name z secret, and, for the sake of simplicity, assume $R \overset{\text{def}}{=} \overline{b}z$. In the π-calculus, process P is may-equivalent to process $\overline{b}d$. We want to show that the implementations of P and $\overline{b}d$ are still equivalent when they are put in the low-level model of communications. We manage to prove this under the assumption that public channel p is both *asynchronous* and *noisy*. Thus, the actual implementation also includes a buffer $B \overset{\text{def}}{=} !p(x).\overline{p}x$ and a noise generator $N \overset{\text{def}}{=} !(\nu\, k)\overline{p}\{k\}_k$ for p. Both noise and asynchrony are necessary to prevent the execution context from detecting traffic on p. Let $V = fn(I_P, \overline{b}d, N, B)$. To sum up, we want to show that

$$(\epsilon_V, \epsilon_V) \vdash (I_P \mid N \mid B) \simeq (\overline{b}d \mid N \mid B) \qquad (4)$$

We do this in two steps. First, we prove that $(\epsilon_V, \epsilon_V) \vdash (\overline{b}d \mid N \mid B) \ll (I_P \mid N \mid B)$. This follows from $(\epsilon_V, \epsilon_V) \vdash \overline{b}d \ll I_P$ (the $\overline{b}d$-action on the LHS can be simulated via communication at p and decryption of $\{d\}_{k_c}$ in the RHS) and then applying the congruence rule (C-PAR).

Now, we prove the converse. Let y be any fresh name and let $\sigma \stackrel{\text{def}}{=} \epsilon_V[^{\{d\}_{k_c}}/_y]$. The crucial step is showing that:

$$(\sigma,\sigma) \vdash p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z \ll p(x).\overline{b}d \tag{5}$$

To see this, first note that for any ζ such that $\widetilde{w} \stackrel{\text{def}}{=} (n(\zeta) - \mathit{dom}(\sigma))$ are fresh and such that $\widehat{\zeta\sigma} \neq -$, we have that: $(\sigma[^{\widetilde{w}}/_{\widetilde{w}}], \sigma[^{\widetilde{w}}/_{\widetilde{w}}]) \vdash \mathtt{let}\ z = \mathtt{dec}_{k_c}(\widehat{\zeta\sigma})\ \mathtt{in}\ \overline{b}z \ll \overline{b}d$ (in fact, the only case for $\mathtt{dec}_{k_c}(\widehat{\zeta\sigma}) \neq -$ is when $\widehat{\zeta\sigma} = \{d\}_{k_c}$, which implies that the LHS is equivalent to $\overline{b}d$). Then (5) above follows applying (C-INP). Now, using (C-PAR) and (5) above, we have that:

$$(\sigma,\sigma) \vdash \overline{p}\{d\}_{k_c} \mid (p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z) \ll \overline{p}\{d\}_{k_c} \mid p(x).\overline{b}d \tag{6}$$

Hence, using (C-RES) and then a standard structural law on restriction (i.e. $(\nu\, a)(A \mid B) \equiv ((\nu\, a)A) \mid B$ if $a \notin \mathit{fn}(B)$), we have:

$$\begin{aligned}&(\epsilon_V, \epsilon_V) \vdash \\ &\quad (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z) \ll \\ &\quad (\nu\, k_c)(\overline{p}\{d\}_{k_c} \mid p(x).\overline{b}d) \equiv (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \mid p(x).\overline{b}d.\end{aligned}$$

Now, note that $(\nu\, k_c)(\overline{p}\{d\}_{k_c})$ can be turned into a particle of noise: $(\epsilon_V, \epsilon_V) \vdash (\nu\, k_c)(\overline{p}\{d\}_{k_c}) \simeq (\nu\, k)(\overline{p}\{k\}_k)$. The thesis follows applying (C-PAR) and the following two properties: noise absorption $(\epsilon_V, \epsilon_V) \vdash N \mid (\nu\, k)(\overline{p}\{k\}_k) \simeq N$ (an instance of the structural law $!A \mid A \equiv\, !A$) and $(\epsilon_V, \epsilon_V) \vdash B \mid p(x).\overline{b}d \ll B \mid \overline{b}d$ (an instance of a general law for asynchronous channels), both of which are not peculiar to cryptography. Note that equality (4) does *not* hold for barbed equivalence. $\diamond$

Example 4.3 (ensuring message delivery)
In this example, we shall use recursive definitions of agent constants, of the kind $A \Longleftarrow S$ where A is an agent constant that may appear in the process expression S (these can be taken as primitive — the theory extends smoothly — or can be coded up using replication like in [12]). We also use the shorthand '$\mathtt{let}\ z = \zeta\ \mathtt{in}\ A\ \mathtt{else}\ B$' in place of '$(\mathtt{let}\ z = \zeta\ \mathtt{in}\ A) + \neg(\mathtt{let}\ z = \zeta\ \mathtt{in}\ tt)B$'. This time we consider a more sophisticated implementation scheme for process P, and prove that (under the assumption of fairness embodied by bisimilarity) this scheme guarantees that a message sent on channel c is eventually delivered. Again, we implement c with an asynchronous and noisy public channel p. This time, however, we need a more complex source of noise: $N \stackrel{\text{def}}{=} !(\nu\, k)!\overline{p}\{k\}_k$. Note the difference from the previous example: N can now spawn at any time a process $(\nu\, k)!\overline{p}\{k\}_k$ that emits a constant noise $\{k\}_k$ at p. The buffer B for p is still $B \stackrel{\text{def}}{=} !p(x).\overline{p}x$. The implementation of P is the process

$$\begin{aligned}I_P &\stackrel{\text{def}}{=} (\nu\, k_c)\Big(!\overline{p}\{d\}_{k_c} \mid R\Big) \qquad \text{where} \\ R &\Longleftarrow p(x).\mathtt{let}\ z = \mathtt{dec}_{k_c}(x)\ \mathtt{in}\ \overline{b}z\ \mathtt{else}\ (\overline{p}x \mid R).\end{aligned}$$

Component $!\overline{p}\{d\}_{k_c}$ constantly emits d encrypted under key k_c on p, while R repeatedly tries to decrypt a ciphertext x received on p using k_c: when the decryption succeeds, the cleartext is sent on b. Let $V = \mathit{fn}(I_P, \overline{b}d, B, N)$; we want to prove that:

$$(\epsilon_V, \epsilon_V) \vdash (I_P \mid B \mid N) \approx (\overline{b}d \mid B \mid N).$$

In order to see this, define $\sigma_1 \stackrel{\text{def}}{=} \epsilon_V[^{\{d\}_{k_c}}/_y]$ and $\sigma_2 \stackrel{\text{def}}{=} \epsilon_V[^{\{k\}_k}/_y]$ (y fresh). We first show that

$$\begin{aligned}(\sigma_1, \sigma_2) \vdash T &\stackrel{\text{def}}{=} \overline{p}\{d\}_{k_c} \mid R \mid B \mid N \approx \\ &\overline{b}d \mid !\overline{p}\{k\}_k \mid B \mid N \stackrel{\text{def}}{=} U\end{aligned} \tag{7}$$

from which the thesis will follow by first applying (C-RES) (with $(\nu\, k_c)$ on the LHS and $(\nu\, k)$ on the RHS) and then the structural law $N \mid (\nu\, k)!\overline{p}\{k\}_k \equiv N$. To prove (7), we consider a relation $\mathcal{R}$ consisting of *two* pairs: $\mathcal{R} = \{\ (\sigma_1 \rhd T\ ,\ \sigma_2 \rhd U)\ ,\ (\sigma_1 \rhd \mathbf{0}\ ,\ \sigma_2 \rhd \mathbf{0})\ \}$ and show that $\mathcal{R}$ is a weak bisimulation, up to parallel composition, contraction and structural congruence. As an example, the move of $\sigma_1 \rhd T$ originating from transition $R \xrightarrow{p\ \{d\}_{k_c}} \equiv \overline{b}d$ is matched by $\sigma_2 \rhd U$ up to structural congruence and parallel composition with $B \xrightarrow{p\ \{k\}_k} \overline{p}\{k\}_k \mid B$; due to lack of space, we omit the other cases. $\diamond$

5. Final Remarks and Related Work

We have studied contextual equivalences and relative proof techniques for a variant of the spi-calculus, an extension of the π-calculus introduced by Abadi and Gordon [3]. We have considered a few examples concerning verification of protocol security, which demonstrate how these techniques can be used in practice.

Two papers closely related to our work are [7] and [5]. In [7], Sangiorgi and one of the authors introduce a lts for the typed π-calculus in which the environment's input/output capabilities on names are explicitly described and updated. Here, we use a similar approach to model the environment's knowledge about names and keys.

Abadi and Gordon presents in [5] a bisimulation approach to cryptographic protocols. When comparing two processes P and Q, an *OK frame-theory pair* $(\mathit{fr}, \mathit{th})$ is used to represent the knowledge of P's and Q's environments. A judgment $(\mathit{fr}, \mathit{th}) \vdash M \leftrightarrow N$ is also introduced to express that the meaning of message M to P's environment is the same as the meaning of message N to Q's environment. When matching transitions, this judgment is used to check indistinguishability of messages M and N being exchanged between P and its environment and Q and its environment. In our case, the indistinguishability of M and

N is guaranteed by requiring that matching transitions have the same environment action and take equivalent environments to equivalent environments. This results in a major difference between the work in [5] and ours when considering output transitions. In our case, given an output transition, it is sufficient, like in standard bisimilarity, to check whether one of the output transitions of the other configuration matches it (these output transitions are finitely many, at least for finite control processes). In the case of [5], one must also look for a new frame-theory pair that consistently extends the old one: this might be not completely trivial, as shown in [10], a paper that addresses some of the non-algorithmic aspects of [5]. Another difference is that in [5] there seems to be little or no compositional reasoning (congruence laws) and no obvious way of tailoring the 'up to' techniques to their setting.

The process algebraic approach to cryptographic protocols has also been followed by Schneider [17], that proposes a CSP-based framework for the analysis and verification of authentication protocols. This approach, differently from the one described in this paper, requires explicitly fixing an attacker and carrying out the analysis with that; changing the attacker would require a new analysis.

The relevance of may-testing to the analysis of security properties has been first discussed by Abadi and Gordon in [3]. May-testing was originally introduced for CCS in [9], and subsequently studied for the π-calculus in [6]; in [8] a precise relationship is established for may-testing between the notions of observer and intersection type.

Acknowledgments We would like to thank the anonymous referees for helpful comments. Discussions with Martin Abadi and Andrew Gordon have helped us to improve the paper.

References

[1] M. Abadi. Protection in Programming-Language Translations. *ICALP'98*, *Proceedings* (K.G. Larsen, S. Skyum, G. Winskel, Eds.), *LNCS* 1443, pp.868-883, Springer-Verlag, 1998.

[2] M. Abadi, C. Fournet, G. Gonthier. Secure implementation of channel abstractions. *LICS'98*, IEEE Computer Society Press, pp. 105-116, 1998.

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

[4] M. Abadi, A.D. Gordon. Reasoning about cryptographic protocols in the spi calculus. *CONCUR'97*, *Proceedings* (A. Mazurkiewicz, J. Winkowsky, Eds.), *LNCS* 1243, pp.59-73, Springer-Verlag, 1997.

[5] M. Abadi, A.D. Gordon. A Bisimulation Method for Cryptographic Protocols. *Nordic Journal of Computing*, 5(4):267-303, 1998.

[6] M. Boreale, R. De Nicola. Testing equivalence for mobile processes. *Information and Computation*, 120: 279-303, 1995.

[7] M. Boreale, D. Sangiorgi. Bisimulation in Name-Passing Calculi without Matching. *LICS'98*, IEEE Computer Society Press, pp. 165-175, 1998.

[8] F. Damiani, M. Dezani-Ciancaglini, P. Giannini. A filter model for mobile processes. *Mathematical Structures in Computer Science*, to appear.

[9] R. De Nicola, M.C.B. Hennessy. Testing Equivalence for Processes. *Theoretical Computers Science*, 34:83-133, 1984.

[10] 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.

[11] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *TACAS'96*, *Proceedings* (T. Margaria, B. Steffen, Eds.), *LNCS* 1055, pp. 147-166, Springer-Verlag, 1996.

[12] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[13] R. Milner. The Polyadic π-calculus: a Tutorial. In *Logic and Algebra of Specification* (F.L. Hamer, W. Brauer, H. Schwichtenberg, Eds.), Springer-Verlag, 1993.

[14] R. Milner, J. Parrow, D. Walker. A calculus of mobile processes, (Part I and II). *Information and Computation*, 100:1-77, 1992.

[15] R. Milner, D. Sangiorgi. Barbed Bisimulation. *ICALP'92*, *Proceedings* (W. Kuich, Ed.), *LNCS* 623, pp.685-695, Springer-Verlag, 1992.

[16] D. Sangiorgi. On the Bisimulation Proof Method. *Mathematical Structures in Computer Science*, 8:447-479, 1998.

[17] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(8):743-758, 1998.