Language Based Secure Communication

Michele Bugliesi and Riccardo Focardi
Dipartimento di Informatica
Università Ca' Foscari di Venezia
{bugliesi,focardi}@dsi.unive.it

Abstract

Secure communication in distributed systems is notoriously hard to achieve due to the variety of attacks an adversary can mount, based on message interception, modification, redirection, eavesdropping or, even more subtly, on traffic analysis. In the literature on process calculi, traditional solutions to the problem either draw on low-level cryptographic primitives, as in the spi or applied-pi calculi, or rely on very abstract, and hard-to-implement, mechanisms to hide communication by means of private channels, as in the pi-calculus. A more recent line of research follows a different approach, aimed at identifying security primitives adequate as high-level programming abstractions, and at the same time well-suited for security analysis and verification in adversarial settings. The present paper makes a step further in that direction. We develop a calculus of secure communication based on core abstractions that support concise, high-level programming idioms for distributed, security-sensitive applications, and at the same time are powerful enough to express a full-fledged adversarial setting. Drawing on this calculus, we investigate reasoning methods for security based on the long-established practice by which security properties are defined in terms of behavioral equivalences. We give a co-inductive characterization of behavioral equivalence, in terms of bisimulation, and develop powerful up-to techniques to provide simple co-inductive proofs. We illustrate the adequacy of the model with several security laws for secrecy and authentication.

1 Introduction

Security in pi-calculus and related process algebraic formalisms is based on a conceptually simple but powerful mechanism: communication over private channels. Data can be created and circulated among the components of a system (the principals of a protocol), and the scope rules guarantee that the environment of the system (the attacker of the protocol) cannot access any piece of data that is exchanged over a private channel. The scope rules are thus the basis for security. Unfortunately, however, the security guarantees they convey are often hard to realize in practice, for several reasons some of which we discuss next (the following observations are not ours, but due to [1]).

Protection from traffic analysis. The observational theory of the pi-calculus validates equations like the one displayed below:

$$\nu n(\overline{n}\langle m\rangle \mid n(x).P) \quad \cong \quad P\{m/x\}$$

Mimicking this behavior in an implementation is costly, as it requires communication protocols resilient to attacks based on traffic analysis.

Message delivery. Another problem with the previous equation is that it presupposes that messages sent on private channels will indeed be delivered to the intended recipients (and to no-one else). While that is convenient at the specification level, implementing a corresponding behavior in a low-level environment requires enough redundancy to compensate for the potential of packet loss and message interception.

Forward Secrecy. The use of private channels for secrecy yields behavioral equivalences that are even more problematic. For instance, the equation

$$\nu n(\overline{n}\langle m\rangle \mid n(x).\overline{p}\langle n\rangle) \quad \cong \quad \nu n(\overline{n}\langle m'\rangle \mid n(x).\overline{p}\langle n\rangle)$$

guarantees the secrecy of the exchange on the private channel n, even though n is made public after the exchange. In a distributed environment, where n might be though as an encrypted channel, an adversary may learn the message sent on n by buffering it and waiting until the key associated to n is finally made public.

In sum, scoping is convenient in pi calculus as a mechanism for writing security specifications, but tends to be too abstract (when applied to channels), hence stronger than what can be achieved in practice. In fact, distributed applications often rely on lower-level cryptographic constructs

to enforce security properties like secrecy and authentication, and have weaker barriers available to protect against attacks.

The invention of the spi-calculus [6] was motivated by the desire to fill this gap and give more adequate foundations for network security. The spi calculus introduces explicit constructs for encryption and decryption, an idea later generalized in the applied-pi calculus [2]. In both cases, this approach succeeds in providing a formal basis for the analysis specification of network security applications. In fact, the development of tools such as *ProVerif* [9] has greatly enhanced our ability to reason on the security properties of applied-pi system specifications. On the other hand, such specifications are specifically targeted at cryptographic protocols, hence they tend to be be somewhat low-level, and not always adequate for reasoning on higher, application-level protocols and their security and functional properties.

In the present paper we investigate an alternative approach in the attempt to strike a new balance between the formal simplicity deriving from high-level abstractions and the flexibility and expressive power required for the specification and implementation of realistic network applications. Drawing on previous attempts in the literature (e.g. [17, 4, 7]) we isolate a core library of communication abstractions designed around the high-level security guarantees they are meant to convey. We inject these abstractions into a variant of the asynchronous pi-calculus, whose semantics we extend to include reductions that make it possible to reason on attacks based on message interception, forwarding and replication. Remarkably, in the resulting semantics *all* exchanges are observable, as the scoping rules do not guarantee complete hiding for communication over private channels. As a result, the semantics of the new calculus breaks the problematic equations we discussed earlier on, easing the implementation task. At the same time, the abstractions are flexible enough to establish useful and interesting security properties. We characterize behavioral equivalence for the calculus co-inductively, in terms of bisimulation, and exploit that characterization to provide very simple co-inductive proofs of security. We illustrate the adequacy of the model by means of a number of security laws for secrecy and authentication, and one more elaborate example.

Contents. Section 2 gives an overview of the high-level communication abstractions, while Section 3 introduces the corresponding network and opponent models. Section 4 formalizes the semantics of networks and their behavioral theory, in terms of an observational equivalence. Section 5 exemplifies the behavioral theory on some distinguishing equations for communication. Section 6 gives an alternative semantics for networks, based on labelled transitions, and shows that bisimilarity characterizes observational equivalence. Section 7 develops a set of useful proof techniques for bisimilarity, and Section 8 applies such techniques to prove some relevant equations for secrecy and authentication. Section 9 outlines a cryptographic implementation of the abstraction primitives. Section 10 concludes with final remarks and a discussion of related work.

2 Secure Channel Abstractions

We presuppose two countable sets $\mathbf{N}$ and $\mathbf{V}$ of names and variables, respectively, and let $a - q$ range over names, w, x, y, z over variables and t, u, v over $\mathbf{N} \cup \mathbf{V}$ when the distinction does not matter. Names enable communication, but they are intended as *identities* rather than channels. In fact, our channels are more structured than customary in name-passing calculi, as they are associated with the names, or identities, of the two end-points they connect. Thus, for instance $\overline{b}\langle a : \dots \rangle$ indicates an output directed to b and originating from a; dually, $b(a : \dots)$ denotes an input performed by b of a message originated by a. Overall, we provide for various communication modes, conveying different security guarantees and requiring different capabilities.

We define the communication primitives as part of a high-level process calculus, defined below. To ease the notation, we introduce a distinguished name, noted $-$, to stand for an *anonymous* identity, and we let $\underline{a}, \underline{b}, \dots$ range over $\mathbf{N} \cup \{-\}$.

H, K	::=	$\overline{u}\langle \underline{a} : \tilde{v} \rangle^{\circ}$	(Output)
		$a(\underline{v} : \tilde{y})^{\circ}.H$	(Input)
		$\mathbf{0}$	(Null)
		$H \mid K$	(Parallel)
		$\mathsf{if}\ u = v\ \mathsf{then}\ H\ \mathsf{else}\ K$	(Conditional)
		$A\langle \tilde{u} \rangle$	(Definition)
		$(\boldsymbol{\nu} a)K$	(Restriction)

Most of the productions are largely standard. The process forms for $\mathbf{0}$, parallel composition $H|K$ and conditional $\mathsf{if}\ u = v\ \mathsf{then}\ H\ \mathsf{else}\ K$ are just as in the pi-calculus. $A\langle \tilde{u} \rangle$ denotes a process defined via a (possibly recursive) definition $A(\tilde{x}) \stackrel{\text{def}}{=} P$, where $\tilde{x}$ contains all the variables that appear free in P, $|\tilde{u}| = |\tilde{x}|$ and A may only occur guarded in P; when $\tilde{x}$ is the empty tuple we write A in place of both $A()$ and $A\langle\rangle$. The restriction $(\boldsymbol{\nu} a)H$ has the familiar pi-calculus syntax but weaker scoping rules due to a different interaction with the synchronization rules (see below).

The remaining productions define the primitives for remote communication. Following [17], our calculus encompasses two fundamental mechanisms for security, based on secrecy and authentication, and includes the communication modes that result from their possible combinations. The secrecy mode is indicated by the symbol $\circ$: secret when $\circ$ is $\bullet$, plain when $\circ$ is missing. The authentication mode, in turn, is signaled by the presence of a distinguished

identity that specifies the source of the message exchanged; the identity is anonymous in case the transmission is non-authentic. Below, we give the intuitions on the semantics of the primitives and modes leaving the formal definitions to Section 4. In particular,

- $\overline{u}\langle - : \tilde{v}\rangle$ denotes a *plain* output, a communication primitive that conveys no security guarantee;
- $\overline{u}\langle a : \tilde{v}\rangle$ denotes a public, but *authentic* output, which provides the receiver with a guarantee on the origin of the message, and ensures that the message cannot be replayed;
- $\overline{u}\langle - : \tilde{v}\rangle^{\bullet}$ denotes a *secret* transmission, providing guarantees that only the intended receiver will be exposed to the message payload: an intruder may become aware of the existence of an output, but not of the message contents;
- finally, $\overline{u}\langle a : \tilde{v}\rangle^{\bullet}$ denotes a *secure* transmission, combining the guarantees of the authentic and secret modes.

In sum, the various output modes protect from message disclosure, replication and forging. On the other hand, an opponent may intercept all outputs, and then selectively forward them back to the network.

The input forms have dual interpretations.

- $a(u : \tilde{y})^{\circ}.H$ denotes an authentic input, which consumes a message sent on a from u, binding $\tilde{y}$ to the tuple of names that form the payload. The input prefix is a binder for the variables $\tilde{y}$, whose scope is the continuation H: instead, u must be instantiated at the time the input prefix is ready to fire. As in the output process, $\circ$ signals the secrecy mode: a public authentic input, noted $a(u : \tilde{y}).P$, synchronizes only with public authentic outputs, whereas secure inputs such as $a(u : \tilde{y})^{\bullet}.P$ only synchronize with secure outputs.
- $a(- : \tilde{y})^{\circ}.H$ is the corresponding input for the non-authentic mode. It consumes a message sent on a, binding the payload to $\tilde{y}$: as in the authentic mode, a non authentic input only synchronizes with a non-authentic output, of the same secrecy mode (as signaled by $\circ$).

As in some dialects of the pi-calculus, notably in the *local* pi-calculus [16], we make a clear distinction between the input and output capabilities for communication, and we disallow the transmission of the former. Note, to this regard, that input prefixes are built around names (not variables) in the channel position. Similarly, we are careful in requiring the use of names (again, not variables) in the sender position of authentic messages. Taken together, the syntactic constraints imposed on the input and output primitives guarantees that a process H never gets to dynamically *impersonate* a new identity, in the sense defined below.

Definition 2.1 (Impersonation)**.** *We say that a process H impersonates an identity a iff H uses a as the subject of an input, as in $a(\underline{u} : \tilde{y})^{\circ}.H$, or as the source of an authentic (public or secret) output, as in $\overline{u}\langle a : \tilde{v}\rangle^{\circ}$.*

Notice that disallowing the transmission of the input capabilities makes it impossible for an adversary to block messages directed to any restricted identity (being secret, the adversary may not input on those identities). To compensate for that, in the next section we introduce a special, low-level primitive, available to the opponent, that provides the capability to intercept any message sent on the network, independently of the scope of the identities of the originator and/or the recipient.

We now put forward an example that illustrates the calculus at work on the specification of a simple protocol for establishing a session between two communication parties. The specification is given by the following definitions:

$$\begin{aligned} D(m) &\stackrel{\text{def}}{=} (A(m) \mid B) \\ A(y) &\stackrel{\text{def}}{=} (\boldsymbol{\nu} k)(\overline{b}\langle a : k\rangle \mid a(b : x).\overline{x}\langle k : y\rangle^{\bullet}) \\ B &\stackrel{\text{def}}{=} (\boldsymbol{\nu} h)b(a : y).(\overline{a}\langle b : h\rangle \mid h(y : z)^{\bullet}.H(z)) \end{aligned}$$

The two parties, A and B, exchange two fresh names, h and k, that are subsequently used for a secret and authentic exchange of the message m. The two fresh names h and k are thus employed to establish a new session between A and B: we will return to this example in Section 8, to give proofs of the security guarantees it conveys on the secrecy and authentication of the message m.

3 Network Abstractions

The high-level calculus provides an idealized set of primitives for programming secure distributed interactions among mutually trusted peers. While that is convenient for programming, an analysis of the security guarantees conveyed by the primitives must necessarily be tested against/conducted within adversarial settings in which an opponent is given full control of the underlying network. To account for that, we introduce a more concrete calculus, of *Networks*, that provides all the low-level capabilities that must be assumed available to the adversary[1]. The produc-

[1]Previous experiments in the literature have explored two directions: either provide for an explicit formalization of the attacker by way of cryptographic calculus (as in the spi/applied pi calculus), or formalize the attacker implicitly [7]. Here we explore an alternative solution: we give an explicit formalization of the attacker, but without resorting to explicit (formal or computational) cryptography

tions of the low-level calculus are defined below.

$$
\begin{array}{rll}
M, N ::= & \overline{u}\langle \underline{a} : \tilde{v} \parallel \tilde{t} \rangle^{\circ} & \text{(Low Output)} \\
\mid & a(\underline{u} : \tilde{y} \parallel \tilde{z})^{\circ}.M & \text{(Low Input)} \\
\mid & \mathbf{0} \mid M|N \mid A\langle \tilde{u} \rangle \mid (\boldsymbol{\nu} a)N \mid \mathsf{if}\ u{=}v\ \mathsf{then}\ M\ \mathsf{else}\ N & \\
\mid & ?z(x : \tilde{y} \parallel \tilde{w}).M & \text{(Intercept)} \\
\mid & !\overline{u}\langle \tilde{v} \rangle.M & \text{(Forward/Replay)}
\end{array}
$$

Before discussing the new syntactic forms, an important remark must be made about the use of names and identities. When reasoning at the network level, it is convenient (and at all reasonable) to have precise ways to include the trusted components of a network and to tell them apart from the adversarial components. To help formalize that distinction, we make a further assumption of the set of names $\mathbf{N}$ and partition it into two sets $\mathbf{N}_t$ and $\mathbf{N}_u$ that identify the *trusted* and *untrusted* identities, respectively. We also assume that α-renaming respects this partition, so that bound names drawn from either set may only be renamed within that set.

We move on with a commentary of the the network specific constructs.

3.1 Input and output forms

The first two productions introduce the network-level primitives for input and output. They are best understood as the low-level counterpart of the programming-level abstractions defined in Section 2. More precisely, $\overline{u}\langle \underline{a} : \tilde{v} \parallel \tilde{t} \rangle^{\circ}$ denotes a (secret or plain, depending on $\circ$) output of $\tilde{v}$ directed to u and originated from $\underline{a}$: the message is authentic from a if $\underline{a} = a$, otherwise no assumption can be made about its actual source. The novelty with respect to the high-level calculus is the $\tilde{t}$ component of the message, which represents the network-level view of the message payload (more on this below). The input prefix $a(\underline{u} : \tilde{y} \parallel \tilde{z}).M$ has a dual interpretation: it binds the variables $\tilde{y}$ and $\tilde{z}$ to the payload and its network-level view, respectively. As in the high-level calculus, the identity component $\underline{u}$ must be instantiated (to '$-$' or a proper identity) at the time the prefix is ready to fire. Similarly, the required use of names in the source position of authentic outputs and in the channel position of inputs rules out undesired impersonations (the notion of impersonation from Definition 2.1) extends as expected to the new calculus).

To motivate the format of the network-level messages, and specifically the role of the $\tilde{t}$ component, we introduce the following definition that connects the high-level abstractions of Section 2 with their network-level counterparts: we call the latter, the *trusted network processes*, or *trusted processes* for short.

Definition 3.1 (Trusted processes)**.** *Given any high-level principal H, we note $[H]$ the corresponding* trusted network process*, i.e. the term that represents H as a network-level process. Below we give the clauses of the mapping $[\cdot]$ for input and output (where $|\tilde{v}| = |\tilde{c}|$ and $|\tilde{x}| = |\tilde{y}|$):*

$$
\begin{array}{lll}
[\overline{u}\langle - : \tilde{v} \rangle] & \triangleq & \overline{u}\langle - : \tilde{v} \parallel \tilde{v} \rangle \\
[\overline{u}\langle a : \tilde{v} \rangle] & \triangleq & (\boldsymbol{\nu} \tilde{c})\overline{u}\langle a : \tilde{v} \parallel \tilde{c} \rangle \\
[\overline{u}\langle \underline{a} : \tilde{v} \rangle^{\bullet}] & \triangleq & (\boldsymbol{\nu} \tilde{c})\overline{u}\langle \underline{a} : \tilde{v} \parallel \tilde{c} \rangle^{\bullet} \\
[b(\underline{u} : \tilde{x})^{\circ}.H] & \triangleq & b(\underline{u} : \tilde{x} \parallel \tilde{y})^{\circ}.[H] \quad (\tilde{y} \cap fv(H) = \emptyset)
\end{array}
$$

The remaining clauses are defined homomorphically. Throughout, we assume that trusted processes only impersonate identities in the set $\mathbf{N}_t$ and only create fresh names in the same set. We reserve the letters P and Q to range over the class of trusted processes, and their run-time derivatives.

As the mapping shows, the network-level view of a message depends on the transmission mode: specifically, it coincides with the payload in plain outputs, while it is a fresh tuple of names that gets associated with each authentic and/or secret output. It is easily seen that this presentation creates an injective map from payloads to their network-level view. For that reason, we often refer to the latter component as the payload (or message) *index*. The chosen format of network-level messages has a natural counterpart in the cryptographic implementation of networks we outline in Section 9. Specifically, it provides an abstraction of the bitstream obtained by the randomized encryption of the payload by which we implement secret outputs, and similarly, an abstraction of the time-variant signature employed to certify the freshness of an authentic transmission. Throughout, we assume that the two components $\tilde{v}$ and $\tilde{t}$ in all messages have the same arity, and that they coincide on public outputs: we make this assumption formal in Definition 3.3.

3.2 Adversarial forms

The last two productions define the primitives that enable an adversary to intercept and subsequently forward, or replay, any message circulating on the network. All network messages may be intercepted by firing the prefix $?z(x : \tilde{y} \parallel \tilde{w}).M$ which is a binder for all its component variables, with scope M: when intercepting the output $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{n} \rangle^{\circ}$, z gets bound to the target b, x to $\underline{a}$ and $\tilde{w}$ to the index $\tilde{n}$. As to $\tilde{y}$, the binding depends on the secrecy mode of the message and on the trust status of the identity b. In particular, if the message is secret and $b \in \mathbf{N}_t$ then $\tilde{y}$ gets bound to $\tilde{n}$, otherwise $\tilde{y}$ is bound to $\tilde{m}$. Thus, intercepting a secret message directed to a trusted principal does not break the secrecy of the payload. Interestingly, a message can be intercepted even if it is directed to a restricted identity, as in $(\boldsymbol{\nu} b)\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{n} \rangle^{\circ}$. Thus the presence of the intercept prefix weakens the effect of the restriction operator substantially, as it makes it impossible to create private/invisible

channels. As such, the intercept prefix is central to our network and opponent models, as it enables observations that are typically available in distributed application, and would otherwise be impossible by means of input.

In addition to this binding effect, intercepting a message also creates a copy of the intercepted message that may subsequently be accessed by the replay/forward form $!\overline{b}\langle\tilde{c}\rangle$ to forward the message back to the network, or produce an arbitrary number of replicas in case the original messages was not authenticated. Notice that this mechanism of forward/replay only depends on the message indexes, and not on the payload.

Throughout the paper, we are interested in reasoning on the behavior of the trusted components of a network in the presence of an adversarial component. Intuitively, the adversarial component may be formed as an arbitrary network process, with the only limitation that it may not impersonate any trusted identity. We define it formally below.

Definition 3.2 (Opponent)**.** *A network process N is adversarial/untrusted iff it only impersonates identities in the set $\mathbf{N}_u$ and only creates fresh names in the same set. An* opponent *is an adversarial/untrusted process.*

Our last definition formalizes the very mild assumption we anticipated on the format of messages in a network.

Definition 3.3 (Well-formed Networks)**.** *We say that plain output $\overline{u}\langle - : \tilde{v} \parallel \tilde{t}\rangle$ is well-formed iff $\tilde{v} = \tilde{t}$; a secret/secure $\overline{u}\langle \underline{a} : \tilde{v} \parallel \tilde{t}\rangle^{\bullet}$ or authentic $\overline{u}\langle a : \tilde{v} \parallel \tilde{t}\rangle$ output is well-formed iff is well-formed iff $|\tilde{v}| = |\tilde{t}|$. A network N is well-formed iff it is closed (has no free variable) and all of its outputs are well-formed.*

Notice that closed trusted processes are well-formed by construction: well-formed networks require opponents to be well-formed as well.

4 Semantics of Networks

To formalize the dynamics of networks, we need further notation and definitions to characterize the semantics of message interception. As we said, intercepting a message also caches a copy of the message: the copy is not directly available as an output message, but may be employed by the adversary to later push the intercepted message back to the network (possibly replicating it).

As a result, to formalize the dynamics of a network we also need a special form to represent the cached messages that arise as a result of an interception. We introduce the new form below as part of what we may call run-time network configurations, defined as follows:

$$M, N \quad ::= \quad \ldots \text{as in Section 3} \ldots \mid \overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*}$$

Now, the dynamics of the calculus is given in terms of reduction and structural congruence, as shown in Table 1. Structural congruence is almost standard, with the exception of the rule (Struct Cache), which states that the duplicates of a non-authentic cached messages may safely be dispensed with. Notice that the equivalence only holds for non-authentic copies: indeed, an inspection of the reduction rules shows that neither authentic messages nor their cached copies are ever duplicated.

The reduction relation follows the intuitions we have given in the previous section. The (Comm) rule is the counterpart of the familiar synchronization rule of the (asynchronous) pi-calculus. The (Intercept) rule, instead, is specific of our calculus and characterizes the semantics of interception. The bindings created upon interception depend on the structure of the intercepted message, as explained earlier on (cf. the side condition in Table 1). In addition this reduction caches a copy of the intercepted message. The remaining axioms, (Forward) and (Replay), provide an adversary with the ability to forward an intercepted message back to the network, and/or to produce new replicas of the message in case the message was not authentic. We note with $\Longrightarrow$ the reflexive and transitive closure of the reduction relation.

Observational Equivalence The behavioral semantics of the calculus is based on a notion of contextual equality defined in terms of reduction barbed congruence [13]. We first define the observation predicate, as usual in terms of barbs.

Definition 4.1 (Barb)**.** *We write $N{\downarrow}b$ whenever $N \equiv (\boldsymbol{\nu}\tilde{n})(\overline{b}\langle \ldots \rangle^{\circ} | N')$ and $b \notin \tilde{n}$*

Based on that, we introduce the following notion of equivalence. We write $M \Downarrow n$ iff $M \Longrightarrow N \downarrow b$ for some N.

Definition 4.2 (Observational Equivalence)**.** *A symmetric relation $\mathscr{R}$ on (run-time) networks is*

- *barb preserving if $M \mathscr{R} N$ and $M \downarrow b$ imply $N \Downarrow b$.*
- *reduction closed if $M \mathscr{R} N$ and $M \longrightarrow M'$ imply $N \Longrightarrow N'$ with $M' \mathscr{R} N'$.*
- *contextual if $M \mathscr{R} N$ implies $M \,|\, O \mathscr{R} N \,|\, O$ for all (closed) opponents O and $(\boldsymbol{\nu}\tilde{n})M \mathscr{R} (\boldsymbol{\nu}\tilde{n})N$ for all names $\tilde{n} \in \mathbf{N}_u$*

Observational equivalence $\cong_O$ is the largest equivalence relation that is reduction closed, barb-preserving and contextual.

Notice that we focus attention to what is sometimes referred to as *static* contexts, thus disregarding closure by guarded forms, conditionals and recursion. While this is becoming a common practice in formalizations of distributed systems (cf. [11]) what is special about our definition is that we

Table 1 Reduction Semantics

Structural congruence

(Struct Par Comm)	$M\|N \equiv N\|M$	
(Struct Par Assoc)	$(N\|N')\|N'' \equiv N\|(N'\|N'')$	
(Struct Par Zero)	$N\|\mathbf{0} \equiv N$	
(Struct Res Zero)	$(\nu a)\mathbf{0} \equiv \mathbf{0}$	
(Struct Res Comm)	$(\nu a)(\nu b)N \equiv (\nu b)(\nu a)N$	
(Struct Res Par)	$M\|(\nu a)N \equiv (\nu a)(M\|N)$	when $a \notin fn(M)$
(Struct Rec)	$A\langle\tilde{w}\rangle \equiv N\{\tilde{w}/\tilde{x}\}$	if $A(\tilde{x}) \stackrel{\text{def}}{=} N$ and $\|\tilde{w}\| = \|\tilde{x}\|$
(Struct If True)	if $a = a$ then M else $N \equiv M$	
(Struct If False)	if $a = b$ then M else $N \equiv N$	when $a \neq b$
(Struct Equiv)	$M \equiv M$, $M \equiv N$ implies $N \equiv M$, $N \equiv N'$ and $N' \equiv N''$ imply $N \equiv N''$	
(Struct Cong)	$N \equiv N'$ implies $(\boldsymbol{\nu} n)N \equiv (\boldsymbol{\nu} n)N'$ and $N \mid N'' \equiv N' \mid N''$	
(Struct Cache)	$\overline{b}\langle - : \tilde{m} \parallel \tilde{n}\rangle^{\circ}_{*} \mid \overline{b}\langle - : \tilde{m} \parallel \tilde{n}\rangle^{\circ}_{*} \equiv \overline{b}\langle - : \tilde{m} \parallel \tilde{n}\rangle^{\circ}_{*}$	

Reduction In the (Intercept) rule, the $\tilde{p}$ are as follows: if $\circ = \bullet$ and $b \in \mathbf{N}_t$ then $\tilde{p} = \tilde{c}$ else $\tilde{p} = \tilde{m}$

(Comm)	$\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid b(\underline{a} : \tilde{y} \parallel \tilde{z})^{\circ}.N$	$\longrightarrow$	$N\{\tilde{m}/\tilde{y},\ \tilde{c}/\tilde{z}\}$
(Intercept)	$\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid ?z(x : \tilde{y} \parallel \tilde{w}).N$	$\longrightarrow$	$\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid N\{b/z,\ \underline{a}/x,\ \tilde{p}/\tilde{y},\ \tilde{c}/\tilde{w}\}$
(Forward)	$\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid !\overline{b}\langle \tilde{c}\rangle.N$	$\longrightarrow$	$\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid N$
(Replay)	$\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid !\overline{b}\langle \tilde{c}\rangle.N$	$\longrightarrow$	$\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid N$

Contextual reduction

(Struct)
$$\frac{M \equiv M' \quad M' \longrightarrow N' \quad N' \equiv N}{M \longrightarrow N}$$

(Res)
$$\frac{N \longrightarrow N'}{(\nu a)N \longrightarrow (\nu a)N'}$$

(Par)
$$\frac{M \longrightarrow M'}{M|N \longrightarrow M'|N}$$

further restrict to *adversarial* contexts. That reflects our initial intention to find a reasoning method specifically targeted at the analysis of security centric properties. As a consequence, we are not interested in a full-blown notion of equivalence, but rather on a notion of equivalence that enables us to analyze the interactions among the trusted principals of a network in terms of the observations that are available to an opponent. When tested on two trusted processes, as in $P \cong_O Q$, that is exactly what our notion of equivalence captures. We discuss this issue further in the next section.

5 Observing Communication

We discuss few equational laws that characterize the behavior of our communication primitives and provide insight into the their distinctive properties. The proofs of some of these equations will be given in Section 8 exploiting the coinductive characterization of behavioral equivalence developed there. By an abuse of notation, we present the equations on the terms of the high-level calculus of Section 2 rather than on their corresponding network processes. In other words we write $H \cong_O K$ as shorthand for $[H] \cong_O [K]$.

The first equation corresponds to the familiar equation of the asynchronous pi-calculus:

$$b(- : \tilde{x}).\overline{b}\langle - : \tilde{x}\rangle \ \cong_O \ \mathbf{0}$$

Just as in the asynchronous pi-calculus, this equation shows that there is no way to test the presence of an input prefix. Here, the equation is also a consequence of the input/output primitives being in plain mode. In fact, it does not extend to the secret version of the forms.

$$b(- : \tilde{x})^{\bullet}.\overline{b}\langle - : \tilde{x}\rangle^{\bullet} \not\cong_O \mathbf{0}$$

The failure is due to the structure of the index of the message exchanged. Indeed, re-packaging the same message in a secret output always gives a different index (corresponding, in an implementation, to the randomness of encrypted messages).

Authentic communication is resistant to impersonation and replay attacks. To illustrate, let $a \in \mathbf{N}_t$ be a trusted identity. The first equation, below, shows that it is never possible to impersonate a: messages expected from a must indeed be sent by a.

$$b(a:x)^\circ.H \cong_O \mathbf{0}$$

The second equation shows that it is never possible to receive two authentic messages from a trusted identity, if that identity has sent just one authentic message.

$$\overline{b}\langle a:m\rangle^\circ \mid b(a:x)^\circ.b(a:y)^\circ.H \cong_O \overline{b}\langle a:\tilde{m}\rangle^\circ \mid b(a:x)^\circ.\mathbf{0}$$

Again, the equation is a consequence of the opponent not being able to impersonate $a \in \mathbf{N}_t$. Similar guarantees hold in more general situations. In particular, a message m sent once by a is always received once, even in the presence of an arbitrary trusted process P that may output other messages as a.

$$\begin{array}{l}(\boldsymbol{\nu} m)(\overline{b}\langle a:m\rangle^\circ \\ \quad \mid b(a:x)^\circ.b(a:y)^\circ.\mathsf{if}\ (x=y=m)\ \mathsf{then}\ H) \mid P \\ \quad \cong_O \\ (\boldsymbol{\nu} m)(\overline{b}\langle a:m\rangle^\circ \\ \quad \mid b(a:x)^\circ.b(a:y)^\circ.\mathbf{0}) \mid P\end{array}$$

Notice, on the other hand, that if the output were non-authentic, the message could be replayed (by an opponent), and H executed, thus breaking the equivalence.

Our next example motivates and clarifies the nature of our notion of contextuality. As we observed, the definition of observational equivalence requires closure under adversarial contexts only: indeed, $\cong_O$ is not in general preserved by composition of equivalent trusted processes. To illustrate, let $b \in \mathbf{N}_t$ be a trusted identity and consider the equation

$$\overline{b}\langle -:m\rangle^\bullet \ \cong_O\ \overline{b}\langle -:m'\rangle^\bullet$$

Once more, given that trusted identities may not be impersonated, the only interaction an opponent may try with the process is by intercepting the messages, but again, since b is trusted, the actual message content is not leaked. On the other hand, given

$$H \stackrel{\text{def}}{=} b(-:x)^\bullet.\mathsf{if}\ x=m\ \mathsf{then}\ \overline{\mathsf{ok}}\langle\rangle\ \mathsf{else}\ \mathbf{0}$$

it is easily seen that

$$\overline{b}\langle -:m\rangle^\bullet \mid H \ \not\cong_O\ \overline{b}\langle -:m'\rangle^\bullet \mid H$$

There is no contradiction here, because H is not an opponent (as it impersonates b, a trusted identity). We thus have a weaker form of compositionality on trusted processes composition, formalized by the following result (cf. Section 6 for a proof sketch).

Theorem 5.1. *Let P, Q be trusted processes. $P \cong_O Q$ implies*

- *$(\boldsymbol{\nu} n)P \cong_O (\boldsymbol{\nu} n)Q$, for all $n \in \mathbf{N}_t$;*
- *$P|R \cong_O Q|R$, for all trusted processes R which do not impersonates identities in $fn(P) \cup fn(Q)$.*

Thus applying the theorem twice on the previous equation, we obtain:

$$(\boldsymbol{\nu} b)(\overline{b}\langle -:m\rangle^\bullet) \mid H \ \cong_O\ (\boldsymbol{\nu} b)(\overline{b}\langle -:m'\rangle^\bullet) \mid H$$

6 LTS and Bisimilarity

We give an alternative, compositional formulation of the semantics of networks, based on labelled transitions. The new semantics constitutes the basis for the proof methods we introduce later in this section, and refine in Section 7.

The labelled transitions are organized in two sets. A first set, in Tables 2 and 3, collects the transitions that correspond to the reduction semantics of Section 4. In most cases the transitions are either standard, or constitute the direct counterpart of the corresponding reductions in Table 1. The two (Output Intercepted) transitions deserve more attention. First notice that in neither transition the label exhibits the secrecy mode: that is to match the semantics of the intercept primitive that synchronizes with all outputs, irrespective of their secrecy. On the other hand, the label does exhibit different information depending on the secrecy mode of the output. Secondly, observe that the transitions leave in their residual a cached copy of the message emitted: this reflects the effect of an interaction with a surrounding context that tests the presence of an output by intercepting it. A further remark is in order on the difference between the two rules that govern scope extrusion. The difference is best understood if we take the view that a channel name comprises the two identities of the end-points it connects: the source and the destination. Under this interpretation the (Open) rule states that the channel name is not extruded, as in the pi-calculus, while the (Open Intercepted) opens the scope in accordance with the reduction semantics by which intercepting a message discloses the identity of the receiver (as well of the sender) even though restricted. The following, standard result connects the reductions with the silent actions in the labelled transition semantics.

Table 2 Labelled Transitions - Input, output, interception and re-emission

(Input)

$$b(\underline{a} : \tilde{y} \parallel \tilde{w})^{\circ}.N \xrightarrow{b(\underline{a}:\tilde{m}\parallel\tilde{c})^{\circ}} N\{\tilde{m}/\tilde{y},\ \tilde{c}/\tilde{w}\}$$

(Output)

$$\overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \xrightarrow{\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle^{\circ}} \mathbf{0}$$

(Open)

$$\frac{N \xrightarrow{(\tilde{p})\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle^{\circ}} N' \quad n \in \{\tilde{m}, \tilde{c}\} - \{b, \underline{a}, \tilde{p}\}}{(\boldsymbol{\nu} n)N \xrightarrow{(n,\tilde{p})\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle^{\circ}} N'}$$

(Open Intercepted)

$$\frac{N \xrightarrow{(\tilde{p})?\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle} N' \quad n \in \{b, \underline{a}, \tilde{m}, \tilde{c}\} - \{\tilde{p}\}}{(\boldsymbol{\nu} n)N \xrightarrow{(n,\tilde{p})?\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle} N'}$$

(Secret Output Intercepted)

$$\frac{b \in \mathbf{N}_t}{\overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\bullet} \xrightarrow{?\overline{b}\langle\underline{a}:\tilde{c}\parallel\tilde{c}\rangle} \overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\bullet}_{*}}$$

(Output Intercepted)

$$\frac{b \notin \mathbf{N}_t \text{ or } \circ \neq \bullet}{\overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \xrightarrow{?\overline{b}\langle\underline{a}:\tilde{m}\parallel\tilde{c}\rangle} \overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*}}$$

Interception and re-emission

(Replay/Forward)

$$!\overline{b}\langle\tilde{c}\rangle.N \xrightarrow{!\overline{b}\langle\tilde{c}\rangle} N$$

(Intercept)

$$?z(x : \tilde{y} \parallel \tilde{w}).N \xrightarrow{?b(\underline{a}:\tilde{p}\parallel\tilde{c})} N\{b/z,\ \underline{a}/x,\ \tilde{p}/\tilde{y},\ \tilde{c}/\tilde{w}\}$$

(Co-forward)

$$\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \xrightarrow{!b(\tilde{c})} \overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}$$

(Co-replay)

$$\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \xrightarrow{!b(\tilde{c})} \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}$$

Lemma 6.1 (Harmony)**.**

- *If* $M \xrightarrow{\alpha} M'$ *and* $M \equiv N$ *then* $N \xrightarrow{\alpha} N'$ *and* $M' \equiv N'$
- $N \longrightarrow N'$ *if and only if* $N \xrightarrow{\tau} \equiv N'$.

The second set of rules is given in Definition 6.2. and provide the observational counterpart of the labelled transitions of Tables 2 and 3. The format of the observational transitions is best understood if we keep in mind that, as we mentioned earlier, we are only interested in characterizing a very specific set of network observations, those available to an opponent. As a result, the observational LTS is obtained by the LTS in Table 2 by filtering away all the transitions that involve the adversarial forms (intercept and forward/reply) as well all the transitions that may not be observed by an opponent by virtue of the restriction the opponent suffers on the use of the trusted identities of a network. In addition (following e.g. [12]) we include a new input transition to capture the observations of an input in an asynchronous setting as the one we have assumed.[2]

[2]We prefer this presentation at this stage as it simplifies the definition of bisimilarity. As in the asynchronous pi calculus, an equivalent presentation results from dropping the additional input transition and relying on asynchronous bisimilarity [8] instead. That presentation, in turn, is useful to control the size of the candidate relations used in equivalence proofs. We take the liberty of using the two formulations interchangeably, favouring the one that best suits our needs in the situation at hand.

Definition 6.2 (Observational LTS)**.** *We say that a network has an observational transition, noted* $N \overset{\alpha}{\longmapsto} N'$*, if and only if it may be derived by the following rules:*

$$\frac{N \xrightarrow{\alpha} N' \quad \alpha \notin \left\{ \begin{array}{l} b(\underline{a} : \tilde{m} \parallel \tilde{c})^{\circ} \mid a \in \mathbf{N}_t \\ (\tilde{p})\overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid b \in \mathbf{N}_t \\ ?b(\underline{a} : \tilde{m} \parallel \tilde{c}), !\overline{b}\langle\tilde{c}\rangle \end{array} \right\}}{N \overset{\alpha}{\longmapsto} N'}$$

$$\frac{\underline{a} = - \quad \textit{or} \quad a \notin \mathbf{N}_t}{N \xmapsto{b(\underline{a}:\tilde{m}\parallel\tilde{c})^{\circ}} N \mid \overline{b}\langle\underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}}$$

Notice that, by definition, $N \overset{\alpha}{\longmapsto} N'$ implies $N \xrightarrow{\alpha} N'$, for $\alpha \neq b(\underline{a} : \tilde{m} \parallel \tilde{c})^{\circ}$, and $N \overset{\tau}{\longmapsto} N'$ iff $N \xrightarrow{\tau} N'$. It is trivial to see that the new transition for $b(\underline{a} : \tilde{m} \parallel \tilde{c})^{\circ}$ does

Table 3 Labelled Transitions - Synchronization and structural rules

(Synch)
$$\frac{M \xrightarrow{(\tilde{p})\overline{\alpha}} M' \quad N \xrightarrow{\alpha} N' \quad \tilde{p} \cap fn(N) = \emptyset}{M \mid N \xrightarrow{\tau} (\boldsymbol{\nu}\tilde{p})(M' \mid N')}$$

(Cond)
$$\frac{(a = b \wedge M \xrightarrow{\alpha} N) \vee (a \neq b \wedge M' \xrightarrow{\alpha} N)}{\mathsf{if}\ a = b\ \mathsf{then}\ M\ \mathsf{else}\ M' \xrightarrow{\alpha} N}$$

(Par)
$$\frac{M \xrightarrow{\alpha} M' \quad bn(\alpha) \cap fn(N) = \emptyset}{M \mid N, N \mid M \xrightarrow{\alpha} M' \mid N, N \mid M'}$$

(Restr)
$$\frac{N \xrightarrow{\alpha} N' \quad n \notin fn(\alpha)}{(\boldsymbol{\nu}n)N \xrightarrow{\alpha} (\boldsymbol{\nu}n)N'}$$

(Rec)
$$\frac{N\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha} N' \quad A(\tilde{x}) \stackrel{\text{def}}{=} N}{A\langle\tilde{w}\rangle \xrightarrow{\alpha} N'}$$

not break the Harmony lemma, and we conclude that that lemma holds for the observational LTS as well.

The notion of bisimilarity arising from the observational LTS transitions is standard.

Definition 6.3 (Network Bisimilarity)**.** *A symmetric relation $\mathscr{R}$ over networks is a weak bisimulation if whenever $M\mathscr{R}N$ and $M \overset{\alpha}{\longmapsto} M'$ with $bn(\alpha) \cap fn(N) = \emptyset$ one has $N \overset{\hat{\alpha}}{\Longmapsto} N'$ and $M'\mathscr{R}N'$. Two networks are bisimilar, written $M \approx N$ iff $M\mathscr{R}N$ for some network bisimulation $\mathscr{R}$.*

A Characterization of Observational Equivalence We move on by studying the relationship between network bisimilarity and observational equivalence. Given the structure of the LTS, the notion of bisimilarity only makes sense when applied to networks that only differ in the trusted components and share a common opponent. That is because the LTS does not look at the opponent transitions, hence it may end-up equating nonequivalent networks. To see that, consider the two opponents $O_1 \triangleq ?z(x : y \parallel w).\overline{b}\langle - : \|\rangle$ and $O_2 \triangleq \mathbf{0}$. Then $O_1 \approx O_2$, as neither has any observational transition, but clearly $O_1 \not\cong_O O_2$. On the other hand, when restricted to trusted processes, bisimilarity coincides with observational equivalence.

Theorem 6.4. $P \cong_O Q$ iff $P \approx Q$, for all P, Q trusted processes.

Proof outline. For the if direction, define the candidate relation $\mathscr{R} = \{(O[P], O[Q]) \mid P, Q \text{ trusted}, O[\cdot] \text{ opponent}\}$ and show that $\mathscr{R}$ is barb preserving, reduction closed and closed by opponent contexts. The proof proceeds by induction on the structure of the context $O[\cdot]$.

For the only if direction the proof follows by coinduction using the relation $\mathscr{R} = \{(P, Q) \mid P \cong_O Q\}$ as candidate. Then proof proceeds by exhibiting the distinguishing context for each transition $P \overset{\alpha}{\longmapsto} P'$. In the general case, the details are notationally costly due to the polyadic nature of the calculus. We show the case when $\alpha = (c)?\overline{b_0}\langle b_1 : b_2 \parallel c\rangle^{\bullet}$ as a simpler representative. In that case, we define:

$$\begin{aligned} O_{\tilde{n}} \;=\; & \overline{\mathsf{ko}}\langle - : \| \rangle \mid \\ & ?x_0(x_1 : x_2 \parallel y).\mathsf{if}\ (x_i = b_i)_{i=0..2}\ \mathsf{then} \\ & \qquad \mathsf{if}\ (y \in \{\tilde{n}\})\ \mathsf{then}\ \mathbf{0} \\ & \qquad \mathsf{else}\ \mathsf{ko}(- : \|).\overline{\mathsf{ok}}\langle - : \ y \parallel y\rangle \\ & \qquad \mathsf{else}\ \mathbf{0} \end{aligned}$$

where the $\tilde{n}$ are the free names in P and Q (collectively), and ko and ok are chosen fresh in $\{c, \tilde{n}\}$. By construction, we have:

$$P \mid O_{\tilde{n}} \Longrightarrow M \equiv (\boldsymbol{\nu}c)(P' \mid \overline{\mathsf{ok}}\langle - : c \parallel c\rangle)$$

Here, as a result of the interception, the opponent caches a copy of the message intercepted, namely $\overline{b_0}\langle b_1 : b_2 \parallel c\rangle^{\bullet}_{*}$. In the labelled transition, this copy is attributed to the derivative of P, that is P', explaining the structure of M.

The rest of the proof is standard. First observe that $M \not\Downarrow \mathsf{ko}$ and $M \Downarrow \mathsf{ok}$. Then, from the hypothesis $P \cong_O Q$, being $O_{\tilde{n}}$ adversarial, we derive $P \mid O_{\tilde{n}} \cong_O Q \mid O_{\tilde{n}}$. Hence there exists N such that $Q \mid O_{\tilde{n}} \Longrightarrow N$ and $N \not\Downarrow \mathsf{ko}$ and $N \Downarrow \mathsf{ok}$. This implies that $O_{\tilde{n}}$ must have consumed the input on ko. Thus, given that ko is fresh to Q, and Q is a trusted process (hence does not intercept/replay), we know that there exists Q' such that $N \equiv (\boldsymbol{\nu}c)(Q' \mid \overline{\mathsf{ok}}\langle - : c \parallel c\rangle)$ and $Q \overset{\alpha}{\Longmapsto} Q'$. Now the proof follows by showing that $M \cong_O N$ implies $P' \cong_O Q'$.

The characterization is useful as it allows us to derive coinductive proofs of observational equivalence. As an illustration, one can now obtain a proof for Theorem 5.1 by showing that $\approx$ is preserved by the contexts given in the statement of the theorem.

7 Proof Techniques

One problem with the coinductive proof technique associated with bisimilarity is that the size of the candidate

relations used in proofs grows easily out of control due to the presence of multiple replicas of the same message. In this section we give two powerful techniques to control that growth.

The two techniques share the common idea to factor out from the candidate relations all pairs of processes containing more than one replica of the same non-authentic messages. As a result, we can restrict to candidates which include the cached copy of each message and at most one replica. Then the two techniques take a further step in dual directions: the first filters out the cached copy as well, whereas the second filters out the replica and keeps the cached copy.

Formalizing the two techniques requires some non trivial technical steps that we outline below. We first introduce notation to treat messages and their cached copies in a more uniform and compact way:

Notation 7.1. *We write* $\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{k}$ *to note the composition* $\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*} \mid \underbrace{\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ} \mid \ldots \mid \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}}_{k}$ *(*$k \geq 0$*). A related notation applies for authentic messages. We write* $\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{0}$ *to denote* $\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{*}$ *and* $\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{1}$ *to denote* $\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle^{\circ}$.

Intuitively, in $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{k}$, k indicates the number of available replicas of a certain message. Notice, to that regard, that for authentic messages, k can only be 0 or 1, as such messages may not be replayed. We denote with $I_b(N)$ the set of tuples that index the messages in the network N directed to b, and are known to the environment:

$$I_b(N) = \{\tilde{c} \mid N \equiv (\boldsymbol{\nu}\tilde{n})(\overline{b}\langle \underline{a} : \tilde{m} \| \tilde{c}\rangle^{\circ}_{k} \mid N') \text{ with } \tilde{c} \cap \tilde{n} = \emptyset\}$$

The condition $\tilde{c} \cap \tilde{n} = \emptyset$ ensures that $\tilde{c}$ is known to the environment, thus the message is either in plain mode or it has been intercepted at least once. (Recall that the non-plain modes generate a fresh $\tilde{c}$ for each output.)

We write P_* to denote a derivative of a trusted process, i.e., $P \stackrel{\gamma}{\Longmapsto} P_*$ and we call it a *run-time trusted process.*

Proposition 7.2. *Let* P_* *be a run-time trusted process. If* $\tilde{c} \in I_b(P_*)$*, then* $P_* \equiv (\boldsymbol{\nu}\tilde{n})(\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{k} \mid P'_*)$ *where* $\tilde{c} \cap \tilde{n} = \emptyset$*,* $\tilde{c} \notin I_b(P'_*)$ *and* $k = 0, 1$ *when* $\underline{a} = a$*.*

The condition $\tilde{c} \notin I_b(P'_*)$ guarantees that there are no other (possibly cached) outputs to b with the same $\tilde{c}$. Intuitively, all the cached and non-cached outputs with the same b and $\tilde{c}$ necessarily contain the same messages $\tilde{m}$.

Based on this result, we introduce an observational transition for a new label, noted $!b(\tilde{c})_k$, which makes the number k of replicas for the message indexed by $\tilde{c}$ observable.

$$\frac{P_* \stackrel{!b(\tilde{c})}{\longmapsto} P'_* \qquad \begin{array}{r} P_* \equiv (\boldsymbol{\nu}\tilde{n})(\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle^{\circ}_{k} \mid Q_*) \\ \tilde{c} \cap \tilde{n} = \emptyset, \tilde{c} \notin I_b(Q_*) \end{array}}{P_* \stackrel{!b(\tilde{c})_k}{\longmapsto} P'_*}$$

We now give new notions of bisimilarity, which only relate run-time trusted processes with at most k copies of the same message, called k-processes:

Definition 7.3 (k-processes)**.** *Let* P_* *be a run-time trusted process. We say that* P_* *is a* k-process*, with* $k \geq 0$*, iff* $P_* \stackrel{!b(\tilde{c})_{k'}}{\longmapsto}$ *implies* $k' \leq k$*.*

7.1 k-Bisimilarity and up-to techniques

Definition 7.4 (k-bisimilarity)**.** *A symmetric relation* $\mathscr{R}_k$ *over* k*-processes is a weak* k*-bisimulation if whenever* $P_* \mathscr{R}_k Q_*$ *and* $P_* \stackrel{\alpha}{\longmapsto} P'_*$ *with* $bn(\alpha) \cap fn(Q_*) = \emptyset$ *and* $\alpha \neq !b(\tilde{c})$*, one has*

- *if* $\alpha = !b(\tilde{c})_k$ *then* $Q_* \stackrel{\alpha}{\longmapsto}$
- *if* $\alpha \neq !b(\tilde{c})_k$ *then* $Q_* \stackrel{\widehat{\alpha}}{\Longmapsto} Q'_*$ *and* $P'_* \mathscr{R}_k Q'_*$*.*

Two k*-processes are* k*-bisimilar, written* $P_* \approx_k Q_*$ *if and only if* $P_* \mathscr{R}_k Q_*$ *for some* k*-bisimulation* $\mathscr{R}_k$*.*

Intuitively, a k-bisimulation is like a standard weak bisimulation for actions which are different from $!b(\tilde{c})_k$. For these latter actions, instead, it requires that they are simulated in a strong way, but on the other hand it disregards the $(k+1)$-process reached by the transition.

The next proposition gives a key property that relates different k-bisimulations.

Proposition 7.5. *If* $P_* \approx_k Q_*$ *with* $k \geq 1$ *then we have* $P_* \approx_{k+1} Q_*$*. Moreover, if* $P_* \approx_k Q_*$ *and* $P_* \stackrel{!b(\tilde{c})_k}{\longmapsto} P'_*$*,* $Q_* \stackrel{!b(\tilde{c})_k}{\longmapsto} Q'_*$ *then* $P'_* \approx_{k+1} Q'_*$*.*

From this proposition, we can prove that bisimilarity is the union of all the k-bisimilarities.

Theorem 7.6. $\approx = \bigcup_{k \geq 1} \approx_k$

Proof. For the inclusion $\supseteq$ the proof is by coinduction. Define $\mathscr{R} = \bigcup_{k \geq 1} \approx_k$. Then, from Proposition 7.5 we know that $\mathscr{R}$ is a bisimulation, hence the desired inclusion. The reverse inclusion derives from the following property: if $P_* \approx Q_*$ then $P_* \stackrel{!b(\tilde{c})_k}{\longmapsto}$ iff $Q_* \stackrel{!b(\tilde{c})_k}{\longmapsto}$ for all $k \geq 0$. This latter property is proved by induction on k.

As a result, we can restrict to 1-bisimilarity as a proof technique, thus limiting the number of replicas in the candidate bisimulations to just one.

Corollary 7.7. *If* $P_* \approx_1 Q_*$ *then* $P_* \approx Q_*$*.*

While relying on 1-bisimilarity is much more convenient than relying on bisimulations, we still have to work with candidates (1-bisimulations) that include replicas in both

the forms $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle_0^\circ$ and $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle_1^\circ$. We will now prove that it is sufficient to keep either one, but not both: the choice of which one determines a different proof technique.

7.2 Bisimulation up-to forward and replay

The first proof techniques shows that we can disregard replicas in the form $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle_0^\circ$. For non-authentic outputs, that implies that in our candidate 1-bisimulations we can refrain from observing the states (i.e. from including the processes) that only store the cached copy of an intercepted message, as the cached copy alone does not give any extra transition that cannot be also performed by $\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle_1^\circ$. Similarly, for authentic messages, it suffices to keep the authentic output $\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle_1^\circ$, disregarding the cached copy.

We introduce a family of relations, indexed by the prefixes $!b(\tilde{c})$, to relate processes with the processes that result from firing the cached copies of the messages indexed by $\tilde{c}$, whenever possible. In particular, we write $P_* >_{!b(\tilde{c})} P'_*$ whenever $P_* \stackrel{!b(\tilde{c})_0}{\longmapsto} P'_*$, or $P'_* \equiv P_*$ if $P_* \stackrel{!b(\tilde{c})_0}{\not\longmapsto}$. Notice that in this latter case P_* is not moving, and captures the impossibility of performing a forward/replay action. We also write $P'_* <_{!b(\tilde{c})} P_*$ whenever $P_* >_{!b(\tilde{c})} P'_*$.

Definition 7.8 (1-bisimilarity up-to forward and replay)**.** *A symmetric relation $\mathscr{R}$ over 1-processes is a weak 1-bisimulation up-to forward and replay if whenever $P_* \mathscr{R} Q_*$ and $P_* \stackrel{\alpha}{\longmapsto} P'_*$ with $bn(\alpha) \cap fn(Q_*) = \emptyset$ and $\alpha \neq !b(\tilde{c})$, one has*

- *if $\alpha = !b(\tilde{c})_1$ then $Q_* \stackrel{\alpha}{\longmapsto}$*
- *otherwise $Q_* \stackrel{\widehat{\alpha}}{\Longmapsto} Q'_*$ and $P'_* >_{!b(\tilde{c})} \mathscr{R} <_{!b(\tilde{c})} Q'_*$.*

Theorem 7.9. *If $\mathscr{R}$ is a weak 1-bisimulation up-to forward/replay then $>_{!b(\tilde{c})} \mathscr{R} <_{!b(\tilde{c})}$ is a 1-bisimulation.*

Corollary 7.10. *If $P_* \mathscr{R} Q_*$ for some weak 1-bisimulation up-to forward/replay, then $P_* \approx Q_*$.*

7.3 k-*Bisimilarity

The second proof technique is dual, and yields candidates which only include $\overline{b}\langle \underline{a} : \tilde{m} \parallel \tilde{c}\rangle_0^\circ$ replicas. The technique arises from an alternative definition of k-bisimilarity, called k-*bisimilarity, that differs from k-bisimilarity in that it imposes a further condition on the processes reached by a $!b(\tilde{c})_k$: namely, whenever a strong transition takes either process back into the class of the k-processes, the other process has a weak matching transition. The definition is as follows:

Definition 7.11 (k-*bisimilarity)**.** *A symmetric relation $\mathscr{R}_k$ over k-processes is a weak k-*bisimulation if whenever $P_* \mathscr{R}_k Q_*$ and $P_* \stackrel{\alpha}{\longmapsto} P'_*$ with $bn(\alpha) \cap fn(Q_*) = \emptyset$ and $\alpha \neq !b(\tilde{c})$, one has*

- *if $\alpha = !b(\tilde{c})_k$ then $Q_* \stackrel{\alpha}{\longmapsto} Q'_*$ for some Q'_*, and for all k-processes P''_* such that $P'_* \stackrel{\gamma}{\longmapsto} P''_*$, there exists a k-process Q''_* such that $Q'_* \stackrel{\widehat{\gamma}}{\Longmapsto} Q''_*$ and $P''_* \mathscr{R} Q''_*$;*
- *if $\alpha \neq !b(\tilde{c})_k$ then $Q_* \stackrel{\widehat{\alpha}}{\Longmapsto} Q'_*$ and $P'_* \mathscr{R}_k Q'_*$.*

*Two k-processes are k-*bisimilar, written $P_* \approx^k_* Q_*$ iff $P_* \mathscr{R}_k Q_*$ for some k-*bisimulation $\mathscr{R}_k$.*

The main result that we outlined for k-bisimilarity, notably, Proposition 7.5, Theorem 7.6 and Corollary 7.7 extend to k-*bisimilarity. Interestingly, Proposition 7.5 holds for $k = 0$ as well when applied to $\approx^k_*$. This implies that 0-*bisimulations (that we call *bisimulations for short) can be used as candidates. That is very convenient for proofs involving non-authentic messages, as *bisimulations are very compact. On the other hand, their use in proofs is slightly more complicate, as proving a relation to be a *bisimulation takes a little more work than proving the corresponding relation to be a 1-bisimulation up-to forward/reply.

8 Secrecy and Authentication Proofs

We show our proof methods at work on a series of security-related observational (in)equalities. To ease the notation, we express our candidates as asynchronous bisimulations (thus disregarding the vacuous input transitions of the observational LTS). As in Section 5, we write $H \cong_O K$ to mean $[H] \cong_O [K]$ and $H \approx K$ to mean $[H] \approx [K]$.

Our first observation is on the role of hidden channels. As we noted, restricting the destination of an output does not hide the presence of the output to an observer. In other words, $(\boldsymbol{\nu} b)\overline{b}\langle \underline{a} : m\rangle^\bullet \not\cong_O \mathbf{0}$, meaning that outputs are always observable, even when they are secret. On the other hand, secret outputs on restricted (or more generally trusted) channels do guarantee the privacy of the payload. This is expressed by the equation we discussed back in Section 5.

$$(\boldsymbol{\nu} b)\overline{b}\langle \underline{a} : m\rangle^\bullet \cong_O (\boldsymbol{\nu} b)\overline{b}\langle \underline{a} : m'\rangle^\bullet \qquad (1)$$

The equality is proved directly by coinduction, using the following candidate, where P and Q are the trusted processes representing the two high-level principals in the equation.

$$\mathscr{R}_{(1)} = \{(P, Q),\ (\overline{b}\langle \underline{a} : m \parallel c\rangle_1^\bullet, \overline{b}\langle \underline{a} : m' \parallel c\rangle_1^\bullet)\}$$

It is easy to verify that $\mathscr{R}_{(1)}$ is a weak 1-bisimulation, up-to forward/replay. Hence $P \approx Q$ as desired. Notice that the

equation holds even without restriction, assumed that b is a trusted identity, i.e., $b \in \mathbf{N}_t$:

$$\overline{b}\langle \underline{a} : m\rangle^{\bullet} \cong_O \overline{b}\langle \underline{a} : m'\rangle^{\bullet}$$

A further variant of equation (1) uses a fresh message to masquerade for m in place of m':

$$(\boldsymbol{\nu} b)\overline{b}\langle \underline{a} : m\rangle^{\bullet} \cong_O (\boldsymbol{\nu} b)(\boldsymbol{\nu} d)\overline{b}\langle \underline{a} : d\rangle^{\bullet} \qquad (2)$$

The proof of this equation is essentially the same as the one we have just outlined. Interestingly, in the non-authentic case, the equation also holds if the output on the right-hand side is in plain mode, as its interception would produce the label $(b, d)?\overline{b}\langle - : d \parallel d\rangle$, which is the same as the one produced by the interception of secret trusted outputs.

$$(\boldsymbol{\nu} b)\overline{b}\langle - : m\rangle^{\bullet} \cong_O (\boldsymbol{\nu} b)(\boldsymbol{\nu} d)\overline{b}\langle - : d\rangle$$

Intuitively, a secret output is indistinguishable by the output of a new (random) name in plain mode.

The equation breaks for the authentic case, since we would have the label $(b, c, d)?\overline{b}\langle a : d \parallel c\rangle$, representing a new name d with its time-variant signature c. As expected, this redundancy makes the message distinguishable from a secret output.

In [6], the spi-calculus characterization of secrecy is given by means of a related equation, that we may express as follows:

$$(\boldsymbol{\nu} b)(\overline{b}\langle \underline{a} : m\rangle^{\bullet} \mid b(\underline{a} : x)^{\bullet}.H(x)) \cong_O (\boldsymbol{\nu} b)(\overline{b}\langle \underline{a} : m'\rangle^{\bullet} \mid b(\underline{a} : x)^{\bullet}.H(x)) \qquad (3)$$

which holds just in case $H(m) \cong_O H(m')$. The proof is similar to the proof of equation (1). Here we use the following candidate and show that it is a *bisimulation.

$$\mathscr{R}_{(3)} = \{ (P, Q),\ (P_*, Q_*) \} \cup \approx^0_*$$

P and Q are the trusted processes corresponding to the high-level principals in the equation, while P_* and Q_* are the residuals of the output intercepted transitions in the two processes, namely $\overline{b}\langle a : m \parallel c\rangle^{\bullet}_* \mid b(a : x)^{\circ}.H(x)$ and $\overline{b}\langle a : m' \parallel c\rangle^{\bullet}_* \mid b(a : x)^{\circ}.H(x)$, respectively. Notice that, since $H(m)$ and $H(m')$ are 0-processes, we have that $H(m) \approx H(m')$ implies $H(m) \approx^0_* H(m')$.

8.1 Authentication

The most basic form of authentication can be stated in terms of the equation $(\boldsymbol{\nu} a)(b(a : x)^{\circ}.P) \cong_O \mathbf{0}$, which may be proved by just observing that neither process has any observational transition.

Again following [6], a more interesting notion of authentication may be formalized by contrasting the system to be authenticated with a system that satisfies the specification trivially. To illustrate, consider the following equation:

$$(\boldsymbol{\nu} a)(\overline{b}\langle a : m\rangle^{\circ} \mid b(a : x)^{\circ}.H(x)) \cong_O (\boldsymbol{\nu} a)(\overline{b}\langle a : m\rangle^{\circ} \mid b(a : x)^{\circ}.H(m)) \qquad (4)$$

Here, by "magically" plugging m in $H(x)$, the equation states that m is the only message that can possibly be received. That is guaranteed because there is just one authentic output in the scope of the restriction. The proof of equation (4) follows co-inductively showing that the following candidate is a *bisimulation:

$$\mathscr{R}_{(4)} = \{ (P, Q),\ (P_*, Q_*) \} \cup \mathbf{Id}$$

Here **Id** is the identity relation, P and Q are the trusted network processes corresponding to the high-level principals in the equation, while P_* and Q_* are the residuals of the output intercepted transitions in the two processes, namely $\overline{b}\langle a : m \parallel c\rangle^{\circ}_* \mid b(a : x)^{\circ}.H(x)$ and $\overline{b}\langle a : m \parallel c\rangle^{\circ}_* \mid b(a : x)^{\circ}.H(m)$,

8.2 Sessions

We conclude our series of examples proving the authenticity and secrecy of properties of the protocol for establishing a private session we introduced back in Section 2. To reason about authentication, let

$$B^{spec}(z') \stackrel{\text{def}}{=} (\boldsymbol{\nu} h)b(a : y).(\overline{a}\langle b : h\rangle \mid h(y : z)^{\bullet}.H(z'))$$

represent the "ideal" definition of B, which differs from B only in the fact that the received z is ignored and, instead, H gets the parameter z'. In other words, $D^{spec}(m) \stackrel{\text{def}}{=} (A(m)|B^{spec}(m))$ represents a process which always delivers m to $Q(z)$. The protocol properties may then be described as follows.

$$\begin{array}{rl} \text{(authenticity)} & D(m) \cong_O D^{spec}(m) \\ \text{(secrecy)} & D(m) \cong_O D(m') \ \text{ if } H(m) \cong_O H(m') \end{array}$$

The proof can be derived in essentially the same way for both equations: we give the proof for the secrecy equation as representative. While we could reason co-inductively, as for the previous equations, in this case it is more convenient to first show an auxiliary equation. Let:

$$\begin{array}{rcl} A'(y) & \stackrel{\text{def}}{=} & \overline{b}\langle a : k\rangle \mid a(b : x).\overline{x}\langle k : y\rangle^{\bullet} \\ B' & \stackrel{\text{def}}{=} & b(a : y).(\overline{a}\langle b : h\rangle \mid h(y : z)^{\bullet}.H(z)) \end{array}$$

We show, that $A'(m) \mid B \cong_O A'(m') \mid B$. Then the proof of our initial equation derives by compositionality as $A'(m) \mid B \cong_O A'(m') \mid B$ implies

$(\boldsymbol{\nu} h)(\boldsymbol{\nu} k)(A'(m) \mid B) \cong_O (\boldsymbol{\nu} h)(\boldsymbol{\nu} h)(A'(m') \mid B)$ by closure under restriction and hence $A(m)|B \cong_O A(m')|B$ because $A(x) \mid B \equiv (\boldsymbol{\nu} h)(\boldsymbol{\nu} k)(A'(x) \mid B)$.

The proof that $A'(m) \mid B \cong_O A'(m') \mid B$ follows by coinduction, choosing the candidate as follows. First define:

$$
\begin{aligned}
\mathscr{R} \;=\; \{ & \\
& (P(m),\, P(m')), \\
& ([a(b:x).\overline{x}\langle k:m\rangle^{\bullet} \mid \overline{a}\langle b:h\rangle \mid h(k:z)^{\bullet}.H(z)], \\
& \qquad [a(b:x).\overline{x}\langle k:m'\rangle^{\bullet} \mid \overline{a}\langle b:h\rangle \mid h(k:z)^{\bullet}.H(z)]), \\
& ([\overline{h}\langle k:m\rangle^{\bullet} \mid h(k:z)^{\bullet}.H(z)], \\
& \qquad [\overline{h}\langle k:m'\rangle^{\bullet} \mid h(k:z)^{\bullet}.H(z)]) \;\}
\end{aligned}
$$

Now, we define $\mathscr{R}_{sec} = \mathscr{R} \cup \approx_1$ and show that $\mathscr{R}_{sec}$ is a 1-bisimulation up-to forward (there are no replicas here). The proof is routine, noting that some of the pairs that arise from $\mathscr{R}$ in the bisimulation game are contained in $\approx_1$. One such pair is $([H(m)], [H(m')])$. That this pair is in $\approx_1$ derives by the following argument. First notice that the hypothesis $H(m) \cong_O H(m')$ implies $[H(m)] \approx [H(m')]$ by Theorem 6.4, and hence $[H(m)] \approx_1 [H(m')]$ by Theorem 7.6, because $[H]$ is a 1-process whenever H is a high-level principal. Another pair is $([a(b:x).\overline{x}\langle k:m\rangle^{\bullet} \mid B'], [a(b:x).\overline{x}\langle k:m'\rangle^{\bullet} \mid B'])$, which arises from $\mathscr{R}$ via an (Output) transition, and is contained in $\approx_1$ as both processes are stuck.

9 Outline of an Implementation

We conclude our presentation of our communication primitives by outlining a cryptographic framework for their implementation. We keep the discussion informal to focus on the main ideas and intuitions, leaving a detailed formalization to our plans for future work.

The basic mechanisms we presuppose on the underlying cryptographic infrastructure are fairly standard, and based on well-known ISO protocols for secrecy and authentication. Our trusted principals may be implemented so as to ensure that all traffic they originate (and receive) is directed to the network over public channels. To enforce the high-level security guarantees provided by our primitives, these protocols may use time-variant signatures to certify the source of authentic messages, and randomized public-key encryption to protect the secrecy of data when desired.

Far from being entirely realistic, (for instance, it would certainly be more reasonably to rely on shared session keys so to improve performance), with some care, these general mechanisms may be assembled into a working implementation. In several respects, that can be accomplished as in similar attempts made in the literature, e.g. [4, 5, 10]). On the other hand, our primitives do have some specific feature that is worth discussing.

One such feature is that the synchronization rules require a precise match on the secrecy mode for the two partners. In the implementation, this presuppose that messages circulate with enough redundancy to distinguish the structure of ciphertexts from that of plain texts.

A further problem for our implementation is that, given that the communication primitives are untyped, we may not count on a formal distinction between names, serving as data, and identities. This is unfortunate, as data and identities have rather different interpretations in a public key infrastructure like the one we are describing: specifically, a piece of data has no predefined format, whereas an identity is typically represented by it associated public key. A solution to this problem is readily obtained, however, by resorting to a fairly standard typing discipline to enforce the required run-time invariants on the separation between names and identities.

A remark is also in order about authentication. As we noted, an authentic exchange not only certifies the source of the message exchange, but also its (the message's) freshness. In an implementation, one way to accomplish that is by means of timestamps: if the source and the destination of a message share a common clock, the destination may verify the authenticity of a message by checking the sender's signature and the timestamp. In distributed systems, however, the presence of a global clock is hardly realistic, and authentic exchanges are typically realized by challenge-response protocols in which the receiver (initiator) challenges the sender (responder) with a nonce that it expects back with the message to deduce the freshness of the message. The problem with an implementation based on challenge-response protocols is that it requires a different interpretation of the authentic input prefix. In particular, we observe that the equation $[(\boldsymbol{\nu} a)b(a:x).H] \cong \mathbf{0}$ is not validated by a challenge-response implementation of authentication, as the presence of the input process is immediately made apparent by the challenge it poses to the responder.

The problem has an obvious solution however, as we can simply adjust our interpretation of the input prefix, by choosing its network-level presentation appropriately. Indeed, it is enough to modify the mapping $[\cdot]$ in Definition 3.1 to so as to mimic, abstractly, the flow of traffic that can be observed in the underlying protocol. That is accomplished as follows, by emitting a nonce when an authentic input is ready to fire:

$$[b(a:\tilde{x})^{\circ}.H] \;\triangleq\; (\boldsymbol{\nu} n)\overline{a}\langle -:n \parallel n\rangle \mid b(a:\tilde{x} \parallel \tilde{y}).[H]$$

The authentication proofs given in Section 8 are still valid under this definition. Of course, authentication has now a slightly more concrete characterization, as with the new definition we have $[(\boldsymbol{\nu} a)b(a:x).H] \cong (\boldsymbol{\nu} n)\overline{a}\langle -:n \parallel n\rangle$ rather than $[(\boldsymbol{\nu} a)b(a:x).H] \cong \mathbf{0}$.

10 Conclusions

We have investigated a new set of security abstractions for distributed communication. The resulting primitives can be understood as a kernel API (Application Programming Interface) for the development of distributed applications. The API primitives are purposely defined without explicit reference to an implementation; at the same time, however, they are designed to be amenable to cryptographic implementation as the one we have outlined.

Certainly, for programming/specifying realistic examples and applications, one would need reliable communications within protected environments (a.k.a. secret channels à la pi-calculus). We do not see any problem in accommodating that feature within our present framework.

The semantic theory and the proof techniques we have developed make the API a convenient tool for the analysis of security-sensitive applications. In its present form, our framework is targeted at (and we argue, well-suited for) secrecy and authentication. Future work includes expending it to account for advanced properties, like anonymity, required in modern network applications such as electronic voting.

Various papers in the literature have inspired or are related to our present approach. A localized use of names, introduced in the Local pi-calculus [15] is discussed and employed in [5] for purposes similar to ours, while the handling of principals and authentication we adopted in the present paper is reminiscent of that in [4].

Other papers with related design are [3, 14, 7]. Of these, the closest to our approach is [7]. While we share some of the initial motivations and ideas, specifically the idea that the environment can mediate all communications, the two target complementary objectives, and differ for a number of design choices and technical results. A first important difference is in the choice of the communication primitives and their semantics: while we accommodate various communication modes, the semantics of communication in [7] makes it possible to only express (what corresponds to) our *secure* communications. As a result, our calculus makes it possible to express a wider range of protocols. A second important difference is that we allow dynamic creation of new principal identities, thus making it possible to express sessions, a feature that is not easily accounted for in [7].

Acknowledgments Work partially supported by M.I.U.R. (Italian Ministry of University and Research) under contract n. 2005015785. Thanks to Pierpaolo Degano and Cedric Fournet for their comments and constructive criticism on an earlier version of the present paper. We also gratefully acknowledge comments from the anonymous referees of the submitted version.

References

[1] M. Abadi. Protection in programming-language translations. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 868–883. Springer, 1998.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, 2001.

[3] M. Abadi and C. Fournet. Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004.

[4] M. Abadi, C. Fournet, and G. Gonthier. Authentication primitives and their compilation. In *POPL'00*, pages 302–315, 2000.

[5] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. *Inf. Comput.*, 174(1):37–83, 2002.

[6] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

[7] P. Adão and C. Fournet. Cryptographically sound implementations for communicating processes. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2006.

[8] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. *Theor. Comput. Sci.*, 195(2):291–324, 1998.

[9] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW*, pages 82–96. IEEE Computer Society, 2001.

[10] M. Bugliesi and M. Giunti. Secure implementations of typed channel abstractions. In M. Hofmann and M. Felleisen, editors, *POPL*, pages 251–262. ACM, 2007.

[11] M. Hennessy. *A Distributed PI-Calculus*. Cambridge University Press, 2007.

[12] K. Honda and M. Tokoro. On asynchronous communication semantics. In M. Tokoro, O. Nierstrasz, and P. Wegner, editors, *Object-Based Concurrent Computing*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer, 1991.

[13] K. Honda and N. Yoshida. On reduction-based process semantics. *Theor. Comput. Sci.*, 151(2):437–486, 1995.

[14] P. Laud. Secrecy types for a simulatable cryptographic library. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM Conference on Computer and Communications Security*, pages 26–35. ACM, 2005.

[15] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. In *Proceedings of ICALP 98*, volume 1443 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[16] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14(5):715–767, 2004.

[17] L. van Doorn, M. Abadi, M. Burrows, and E. Wobber. Secure network objects. In *Secure Internet Programming*, pages 395–412, 1999.