Secure implementations of typed channel abstractions

(Extended Abstract)

Michele Bugliesi Marco Giunti
Dipartimento di Informatica
Università Ca' Foscari di Venezia

Abstract

The challenges hidden in the implementation of high-level process calculi into low-level environments are well understood [1]. This paper develops a secure implementation of a typed pi calculus, in which type capabilities are employed to realize the policies for the access to communication channels. Our implementation translates the typed capabilities of the high-level calculi into corresponding term capabilities protected by encryption keys only known to the intended receivers. As such, the implementation is effective even in the presence of open contexts for which no assumption on trust and behavior may be made. Our technique and results draw on, and extend, previous work [3] on secure implementation of channel abstractions in a dialect of the join calculus. In particular, our translation preserves the forward secrecy of communications in calculi which support the dynamic exchange of write *and/or* read access rights among processes. We establish the adequacy and full abstraction of the implementation by contrasting the untyped equivalences of the low-level cryptographic calculus, with the typed equivalences of the high-level source calculus.

1 Introduction

The use of types for resource access control is a long established technique in the literature on process calculi [13], and so is the application of typed equivalences to reason on the behavior of typed processes [10, 14]. Resource control is achieved by predicating the access to communication channels to the possession of certain type capabilities, and by having a static typing system ensure that the resulting policies are complied with by well-typed processes. Typed equational techniques, in turn, draw on judgments of the form $\mathrm{I} \models P \cong Q$ stating the indistinguishableness (hence the equivalence) of two processes, P and Q, in enclosing contexts that have access to the names in the type environment I via the type capabilities assigned to them by I. Typed equations of this kind are very effective whenever we have control on the structure of the contexts observing our processes, i.e., whenever may assume that such contexts are well-typed.

The question we address in this paper is whether the same kind of reasoning can still be relied upon when we extend the class of observing contexts to arbitrary, potentially ill-typed contexts. Stated more explicitly: can we deploy our typed processes as low-level agents to run in distributed environments, in a fully abstract manner, i.e. preserving the typed behavioral congruences we have established? This appears to be an important question, as it constitutes a fundamental prerequisite to the use of typed process calculi as an abstract specification tool for concurrent computations in distributed, open systems.

In [8] we argue that the desired correspondence may hardly be achieved for high-level process calculi relying on static typing alone. The solution we envision in that paper is based on a new typing discipline that combines static and dynamic typing. Specifically, we introduce a typed variant of the pi-calculus in which the output construct, noted $a\langle v@T\rangle$, uses type coercion to enforce the delivery of v at the type T, regardless of the type of the communication channel a. A static typing system guarantees that v has indeed the coercion type T, while a mechanism of dynamically typed synchronization guarantees that v is received only at supertypes of T, so as to guarantee the type soundness of each exchange

By breaking the dependency between the types of the transmission channels and the types of the names transmitted, distinctive of the traditional approaches to typing in the pi and related calculi [13, 10], we can safely reduce the capability types to the simplest, flat structure that only exhibits the read/write access rights on channels, regardless of the types of the values transmitted. Furthermore, and more interestingly for our present concerns, the combination of type coercion and dynamically typed synchronization allow us to gain further control on the interactions among processes and, consequently, on the interaction between processes and their enclosing context. Based on that we are able to recover fully abstract implementations of the high-level specifications, i.e. implementations for which the

typed congruences established for the specifications are true of the low-level implementations. The basic idea is rather simple, and suggested by the very structure of the types. Briefly, we represent a channel with a pair of asymmetric keys, an encryption key to transmit and a decryption key to receive data, and establish the following correspondence between the cryptographic keys and the type-level capabilities: $[\![n@\mathsf{w}]\!] = n^+$ and $[\![n@\mathsf{r}]\!] = n^-$. Then, we rely on the following variant of the standard representation of a communication over a channel in terms of the exchange of encrypted packets over the public network.

$$[\![n\langle m@A\rangle]\!] = net\langle\{[\![m@A]\!]\}_{n^+}\rangle$$
$$[\![n(x).P]\!] = net(y).\mathsf{decrypt}\ y\ \mathsf{as}\ \{x\}_{n^+}\ \mathsf{with}\ n^-\ \mathsf{in}\ [\![P]\!]$$

While this representation is appealing in its simplicity, it suffers from a number of shortcomings, first made explicit by Abadi in [1]. In subsequent work [3], Abadi, Fournet and Gonthier have shown how to counter these problems and recover a full abstract implementation for the join calculus.

The fundamental obstacle against using the solution of [3] for our purposes is related to the so called problem of *forward secrecy*. To illustrate, we use following example adapted from [1]. Let P and Q be the two processes below (where we omit the type coercions whenever irrelevant):

$$P = (\mathsf{new}\ n)(n\langle m\rangle \mid n(x).p\langle n@\mathsf{r}\rangle)$$
$$Q = (\mathsf{new}\ n)(n\langle m'\rangle \mid n(x).p\langle n@\mathsf{r}\rangle)$$

It is not difficult to be convinced that P and Q are behaviorally equivalent (essentially under any typing assumption), as m and m' are sent over a secret channel and no high-level context may recover the content of messages sent. On the other hand, a low-level context may tell $[\![P]\!]$ from $[\![Q]\!]$ by buffering the message sent on n and then deciphering it when n^- is published. In [3], this problem is avoided altogether, as the join calculus does not allow names to be communicated with read capabilities, a feature that instead constitutes one of the fundamental ingredients of our typed calculus. Hence, to recover forward secrecy, we need a more structured representation of type capabilities to make sure that distributing a read capability does not correspond to leaking any decryption key. In the present paper, we show how this can be done in a variant of the applied pi-calculus [2].

The solution is based on the representation of a channel as a process that serves input and output requests, so that each exchange of messages is the result of two separate protocols with writer and reader clients. All channels are associated with two separate key-pairs. The decryption keys are always stored securely at the channel, and never leaked; the encryption keys, in turn, are available to the clients that have read and/or write access to the channel. In the write protocol, the client sends data, and the channel buffers it on private queue; in the read protocol, the client sends a session key and the server returns data encrypted with that key. Publishing a read/write capability on a channel corresponds to publishing the read/write encryption keys associated with the channel.

Under appropriate, mostly standard, hypothesis on the properties of the underlying network we show that a translation based on these ideas is sound. Full abstraction, instead, is harder to achieve as we need to build safeguards against attacks that exploit malformed data or malicious channels that intentionally leak their associated decryption keys. To account for that, we complement the translation with a proxy-service mechanism to ensure that all communication protocols take place via system generated (hence secure) channels. We devise a distributed implementation of the proxy service for a variant of our high-level process calculus in which processes are assigned with different domains, each one providing a local proxy. We prove that the resulting implementation is fully abstract, by showing that the untyped equivalences of the low-level cryptographic calculus coincide, via the translation, with the typed equivalences of the high-level calculus. As a byproduct, since our high-level process calculus is a conservative extension of the untyped pi calculus, we also have a direct fully abstract implementation of the pi calculus.

Plan of the paper. §2 and §3 review our typed pi calculus and the applied pi calculus, respectively. §4 details the implementation we have outlined and establishes the main results. §5 concludes with final remarks.

2 A pi calculus with dynamic typing

Given the intended use of calculus as a specification language for distributed system, we opt for an asynchronous version of the calculus. However, the same results and technique would apply, mutatis mutandis, to the synchronous case.

We presuppose countable sets of names and variables, ranged over by $a-n$ and $x,y,\ldots$ respectively. We use u,v to range over names and variables, when the distinction does not matter. The syntax of processes is given below.

$$\begin{array}{lcll} A,B & ::= & \mathsf{ch}(\mathsf{rw}) \mid \mathsf{ch}(\mathsf{r}) \mid \mathsf{ch}(\mathsf{w}) \mid \top & \\ P,Q & ::= & \mathbf{0} \mid P \mid Q \mid (\mathsf{new}\ n)P \mid\ !P & \textit{pi calculus} \\ & \mid & [u=v]P;Q & \textit{matching} \\ & \mid & u\langle\tilde{v}@\tilde{A}\rangle & \textit{type-coerced output} \\ & \mid & u(\tilde{x}@\tilde{A}).P & \textit{typed input} \end{array}$$

The types $\mathsf{ch}(\cdot)$ are the types of channels, built around the capabilities r, w and rw which provide *read*, *write* and *full*

fledged access to the channel, respectively. To ease the notation, we henceforth denote channel types by only mentioning the associated capabilities. $\top$ is the type of all values and may be used to provide no access to the channel.

The subtyping relation $<:$ is the lattice defined simply as follows (with A any type) $\mathsf{rw} <: \mathsf{r}$, $\mathsf{rw} <: \mathsf{w}$, $A <: \top$, and with join ($\sqcup$) and meet $\sqcap$ operations arising as expected. Type environments, ranged over by Γ, Δ, are finite mappings from names and variables, or names, to types. Following [10], we extend type environments by using the meet operator: $\Gamma \sqcap u : A = \Gamma, u : A$ if $u \notin dom(\Gamma)$, otherwise $\Gamma \sqcap u : A = \Gamma'$ with Γ' differing from Γ only at u: $\Gamma'(u) = \Gamma(u) \sqcap A$. Subtyping is extended to type environments as expected. If $\Gamma(n) <: \mathsf{r}$, we say that $\Gamma^{\mathsf{r}}(n)$ is defined, written $\Gamma^{\mathsf{r}}(n)\downarrow$. $\Gamma^{\mathsf{r}}(n)\uparrow$ indicates that $\Gamma(n)$ is undefined or $\Gamma(n) \not<: \mathsf{r}$. Dual notation is employed for the write capability.

Typing System. The typing rules, in Table 1, are largely standard, but with important specificities. The typing of matching is inherited directly from [10]: as in that case it allows a more liberal typing of the *then* branch, based on the knowledge that the two values tested are indeed the same. To illustrate, as a result of that rule, the following judgment is derivable: $a : \mathsf{r}, b : \mathsf{w} \vdash [a = b]\, a\langle\rangle; \mathbf{0}$. The rules for input and output are distinctive of our typing system. In T-Out@ the value v, written at channel u, is tagged with a type B, with the intention to force its delivery at the type B (or at a supertype). As we show below, dynamic type checks performed upon synchronization ensure that values delivered at a given type will only be received at the same or higher types.

Operational Semantics. The dynamics of the calculus is defined in terms of a labelled transition system built around the actions $\alpha \in \{\tau, u(\tilde{v}@\tilde{A}), (\tilde{c})\, u\langle\tilde{v}@\tilde{A}\rangle\}$. The output transition $(\tilde{c})\, u\langle\tilde{v}@\tilde{A}\rangle$ carries a type tag along with the output value: it represents the output of (a tuple, possibly including fresh) values $\tilde{v}$ at the types $\tilde{A}$; dually, the input action $u(\tilde{v}@\tilde{A})$ represents input of $\tilde{v}$ at the types $\tilde{A}$. As promised synchronization is dynamically typed: complementary labels synchronize if the agree on the type of the value exchanged. As proved discussed in [8], the dynamic check ensure that well-typing is preserved by reduction.

Observational Equivalence. The notion of observational equivalence, based on weak bisimulation, is inherited almost directly from [10]. As usual in typed equivalences, we observe the behavior of processes by means of contexts that have a certain knowledge of the processes, represented by a set of type assumptions contained in a type environment. However, following [10], we take the view that the typing information available to the context may be different (less informative) than the information available to the system. Thus, while the system processes may perform certain action because they posses the required (type) capabilities, the same may not be true of the context. We formalize these intuitions below, following [10].

Given two type environments Γ and I, we say that Γ is compatible with I if and only if $\mathrm{dom}(\Gamma) = \mathrm{dom}(\mathrm{I})$ and $\Gamma <: \mathrm{I}$.

Definition 1. *A type-indexed relation $\mathscr{R}$ is a family of binary relations between processes (hence closed) indexed by type environments. We write $\mathrm{I} \models P \mathscr{R} Q$ to mean that (i) P and Q are related by $\mathscr{R}$ at I and (ii) there exist Γ and Δ compatible with I such that $\Gamma \vdash P$ and $\Delta \vdash Q$. We often write $\mathrm{I} \models P \mathscr{R} Q$ as $P\ \mathscr{R}_{\mathrm{I}}\ Q$.*

Definition 2 (Contextuality). *A type-indexed relation $\mathscr{R}$ is contextual whenever (i) $\mathrm{I} \vDash P \mathscr{R} Q$ implies $\mathrm{I}, a : A \vDash P \mathscr{R} Q$, (ii) $\mathrm{I} \vDash P \mathscr{R} Q$ and $\mathrm{I} \vdash R$ imply $\mathrm{I} \vDash P \,|\, R \mathscr{R} Q \,|\, R$, and (iii) $\mathrm{I}, a : A \vDash P \mathscr{R} Q$ implies $\mathrm{I} \vDash (\mathsf{new}\ a) P \mathscr{R} (\mathsf{new}\ a) Q$.*

Given a type environment I, and a closed process P, we define the barb predicate relative to the environment I as follows. First define $P \downarrow_a$ iff $P \xrightarrow{(\tilde{c})a\langle\tilde{b}@\tilde{B}\rangle}$ and $P \Downarrow_a$ iff $P \Longrightarrow \downarrow_a$, where $\Longrightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$. Then: $\mathrm{I} \vDash P \downarrow_a \triangleq \mathrm{I}(a)^{\mathsf{r}}\downarrow \wedge P \downarrow_a$, and $\mathrm{I} \vDash P \Downarrow_a \triangleq \mathrm{I}(a)^{\mathsf{r}}\downarrow \wedge P \Downarrow_a$.

Definition 3 (Typed behavioral equivalence). *Typed behavioral equivalence, noted $\cong^{\pi}$, is the largest symmetric and contextual type-indexed equivalence relation $\mathscr{R}$ such $\mathrm{I} \models P \mathscr{R} Q$ implies (i) if $\mathrm{I} \vDash P \downarrow_n$ then $\mathrm{I} \vDash Q \Downarrow_n$, and (ii) if $P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'$ and $\mathrm{I} \models P' \mathscr{R} Q'$ for some Q'*

A fundamental difference between our notion of equivalence and that of [10] arises as a consequence of the different typing disciplines. In our system, the rule for typed value exchange guarantees that the context obtains values emitted on a public channel at the static type occurring in the coercion associated with the output. Conversely, in [10], the type at which the context acquires the new name is determined by the type information that the context has on the channel used for output: that, in turn, presupposes that the context "behaves" in that it will not try to acquire from the value emitted more than allowed by the read type of the transmission channel. As a consequence, in [10] reasoning on the access control policies can hardly be carried out without assuming that the context is well-typed, with the same type system. In our solution, instead, an appropriate use of coercion may be put in place to prevent the context from acquiring, upon output by the system, more information than it was intended to. A corresponding mechanism can be realized in terms of a low-level construction in the applied pi calculus. We illustrate how this can be done in §4.

Table 1 Typing and Dynamics for the pi calculus

Typing rules The typing rules for name and variable projection, and for the process forms of parallel composition, replication, restriction and nil are entirely standard cf. [13, 15]). The remaining rules are given below.

(T-Match)
$$\frac{\Gamma \vdash Q \quad \Gamma(u) = A \quad \Gamma(v) = B \quad \Gamma \sqcap u : B \sqcap v : A \vdash P}{\Gamma \vdash [u = v]\, P; Q}$$

(T-Out@)
$$\frac{\Gamma^{\mathrm{w}}(u)\downarrow \quad \Gamma \vdash \tilde{v} : \tilde{B}}{\Gamma \vdash u\langle \tilde{v}@\tilde{B}\rangle}$$

(T-In@)
$$\frac{\Gamma^{\mathrm{r}}(u)\downarrow \quad \Gamma, \tilde{x} : \tilde{A} \vdash P}{\Gamma \vdash u(\tilde{x}@\tilde{A}).P}$$

Labelled Transitions The transitions for matching, parallel composition, restriction and replication are standard. The remaining transitions are given below.

(PI-OUTPUT@)
$$\frac{}{a\langle \tilde{v}@\tilde{B}\rangle \xrightarrow{a\langle \tilde{v}@\tilde{B}\rangle} \mathbf{0}}$$

(PI-INPUT@)
$$\frac{}{a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}}$$

(PI-OPEN@)
$$\frac{P \xrightarrow{(\tilde{c})a\langle \tilde{v}@\tilde{B}\rangle} P' \qquad b \neq a,\ b \in fn(\tilde{v})}{(\mathsf{new}\ b)P \xrightarrow{(b,\,\tilde{c})a\langle \tilde{v}@\tilde{B}\rangle} P'}$$

(PI-CLOSE@)
$$\frac{P \xrightarrow{(\tilde{c})a\langle \tilde{v}@\tilde{B}\rangle} P' \quad Q \xrightarrow{a(\tilde{v}@\tilde{B}')} Q' \quad \tilde{B} <: \tilde{B}' \quad \tilde{c} \cap fn(Q) = \varnothing}{P \,|\, Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{c})(P' \,|\, Q')}$$

3 The applied pi calculus

The applied pi-calculus we use is an asynchronous version of the original calculus of [2], in which we assume that destructors are only used in let-expressions and may not occur in arbitrary terms. This is becoming common practice in the presentations of the applied pi calculus [5, 6, 4]. We review the core calculus here and defer the discussion on the full set of constructors and deconstructors used in the translation to §4.

As for the high-level calculus, we presuppose countable sets of names and variables, under the same notational conventions. In addition the calculus is characterized by a finite set of function symbols Σ from which terms may be formed.As in [6], we distinguish constructors and destructors, and use the former to build terms, the latter in let expressions to take terms apart. Constructors are typically ranged over by f, destructors by d. Terms are built around variables and constructors, expressions correspond to destructor application:

M,N	::=	$a,b,\ldots$	*channel names*
		$x,y,\ldots$	*variables*
		$f(M_1,\ldots,M_n)$	*constructor application*
E	::=	$d(\tilde{M})$	*expression*

A value is a term without variables. We always assume that constructors are applied consistently with their arity. Processes are defined as follows:

$$P,Q, \quad ::= \quad \mathbf{0} \mid M\langle \tilde{N}\rangle \mid M(\tilde{x}).P \mid P \,|\, Q \mid (\mathsf{new}\ n)P \mid\\ !P \mid \mathsf{let}\ x = E\ \mathsf{in}\ P\ \mathsf{else}\ Q$$

Input prefixing, let, and restriction are binders: $M(x).P$ and $\mathsf{let}\ x = E\ \mathsf{in}\ P\ \mathsf{else}\ Q$ bind the variable x in P, $(\mathsf{new}\ n)P$ binds the name n in P. The notions of free and bound names/variables arise as expected. The process $\mathsf{let}\ x = E\ \mathsf{in}\ P\ \mathsf{else}\ Q$ tries to evaluate E; if that succeeds x is bound to the resulting term and the process continues as P (with the substitution in place). Otherwise the process reduces to Q. The evaluation of E is governed by a set of definitions which give semantics to the destructor. Each definition has the form $d(\tilde{M}) \doteq N$ where the terms $\tilde{M}$ and N have no free names and $fv(N) \subseteq fv(\tilde{M})$. Then $d(\tilde{M})$ is defined only if there is a definition $d(\tilde{M}') \doteq N$ and a substitution σ such that $\tilde{M} = \tilde{M}'\sigma$, in which case $d(\tilde{M})$ evaluates to the term $N\sigma = N^*$, noted $d(\tilde{M}) \to N^*$. Conversely, we note $d(\tilde{M}) \not\to$ whenever there is no $\tilde{M}'\sigma = \tilde{M}$ with a defining equation. We say that a constructor f is *one-way* if no destructor application ever returns its argument(s).

We always omit trailing nil processes and similarly write $\mathsf{let}\ x = E\ \mathsf{in}\ P$ instead of $\mathsf{let}\ x = E\ \mathsf{in}\ P\ \mathsf{else}\ \mathbf{0}$. We also use multiple let-bindings instead of writing the corresponding nested definitions. Finally, we define:

$$\mathsf{rec}X.P \triangleq (\mathsf{new}\ a)\ (a\langle\rangle \,|\, !a().P\{a\langle\rangle/X\}) \quad a \notin fn(P)$$

Our applied pi calculus includes destructors to project the elements of tuples (noted π_i, as well as two constructors for lists, :: (cons) and $\emptyset$ (nil), together with the the standard destructors *hd* and *tl*. In addition, we rely upon the constructors *hash*, *ek*, *dk*, *sk*, *pub*, *priv*, *sign*, and *cipher*, and the destructors *equals*, *decipher*, *verify*, defined by the following equations:

$$\begin{array}{rcl} \mathit{equals}(x,x) & \doteq & x \\ \mathit{decipher}(\mathit{cipher}(x,\mathit{ek}(y)),\mathit{dk}(y)) & \doteq & x \\ \mathit{decipher}(\mathit{cipher}(x,\mathit{sk}(y)),\mathit{sk}(y)) & \doteq & x \\ \mathit{verify}(\mathit{sign}(x,\mathit{priv}(y)),\mathit{pub}(y)) & \doteq & x \end{array}$$

The one-way unary constructor *hash* generate a hash from the seed M. The two unary one-way constructors *ek* and *dk* generate encryption and decryption keys $\mathit{ek}(M)$ and $\mathit{dk}(M)$ from a seed M. We often abbreviate $\mathit{ek}(M)$ to M^+ and $\mathit{dk}(M)$ to M^-. A unary one-way constructor *sk* generates a shared key $\mathit{sk}(M)$ from the seed M. The one-way constructor *pub* generate a public key $\mathit{pub}(M)$ from the seed M while *priv* generates a private key $\mathit{priv}(M)$ from M. We often abbreviate $\mathit{pub}(M)$ with M_{ID}. Signatures are built using the binary constructor *sign* and checked by using the destructor *verify*. Encrypted packets are formed around the binary constructor *cipher*, and taken apart by using the destructor *decipher*. We often use the conventional spi-calculus notation $\{\tilde{M}\}_N$ for the encrypted packet $\mathit{cipher}(\tilde{M},N)$; we often overload the notation and write $\{\tilde{M}\}_{\mathit{priv}(N)}$ to indicate the signed packet $\mathit{sign}(\tilde{M},\mathit{priv}(N))$. We define conditionals in terms of lets and *equals* (following [6])

$$\begin{array}{rcl} \mathsf{if}\ M=N\ \mathsf{then}\ P\ \mathsf{else}\ Q & \triangleq & \mathsf{let}\ x=\mathit{equals}(M,N)\ \mathsf{in}\ P \\ & & \mathsf{else}\ Q\ \ (x\notin \mathit{fv}(P,Q)) \end{array}$$

Finally, we introduce an explicit form of decryption to bind multiple variables as in the original spi-calculus:

$$\begin{array}{l} \mathsf{decrypt}\ M\ \mathsf{as}\ \{\tilde{y}\}_N\ \mathsf{in}\ P\ \mathsf{else}\ Q\ \triangleq \\ \quad \mathsf{let}\ x=\mathit{decipher}(M,N)\ \mathsf{in} \qquad (x\notin \mathit{fv}(P,Q)) \\ \quad\quad \mathsf{let}\ y_1=\pi_1(x),\ldots,y_n=\pi_n(x)\ \mathsf{in}\ P\ \mathsf{else}\ Q \end{array}$$

Operational semantics. The operational semantics is defined in terms of labelled transitions. Following an increasingly common practice [6, 4], it does not rely on active substitutions as in the original formulation. The labelled transitions, in Table 2 are standard.

Most of the transitions are standard. For simplicity, we require that all synchronizations occur on channel names, rather then arbitrary terms. The treatment of let is taken from [6]. The transitions are only defined over closed processes (with no free variables). Given and such process P, we define: $P\downarrow_a \triangleq P \xrightarrow{(\tilde{n})a\langle\tilde{M}\rangle}$, $P\Downarrow_a \triangleq P \Longrightarrow\downarrow_a$.

Behavioural equivalence. As in Section 2, we rely on a notion of behavioral equivalence based on weak bisimulation, and relative to contexts with a certain knowledge about names and terms.

A *term environment* ρ is a finite substitution from variables to values. We write $\mathit{fn}(\rho)$ to mean $\mathit{fn}(\mathrm{Range}(\rho))$. Substitutions may only be extended with new bindings for fresh variables: $\rho,M/x$ indicates the extension of ρ with $x\notin\mathrm{dom}(\rho)$. Given a term environment ρ, we let $\mathscr{A}(\rho)$ be the analysis of ρ, that is, the environment obtained by extending ρ with new bindings for the terms resulting from the application of destructors to the range of ρ. Formally:

Definition 4. *The analysis $\mathscr{A}(\rho)$ of ρ is the smallest substitution σ extending ρ that is closed by the following rule:*

$$\frac{d(\tilde{N})\doteq N \quad \tilde{N}\sigma\subseteq \mathrm{Range}(\sigma) \quad (z\notin\mathrm{dom}(\sigma))}{N\sigma/z\in\sigma}$$

Abusing the notation we often write $N\in\mathscr{A}(\rho)$ to mean $N\in\mathrm{Range}(\mathscr{A}(\rho))$. Given process P we say that ρ defines P, written $\rho\vdash P$, if $\mathit{fv}(P)\subseteq\mathrm{dom}(\rho)$ and $\mathit{fn}(P)\cap\mathit{fn}(\rho)=\emptyset$. We use the same convention and notation for terms.

Definition 5 (Term-indexed relation). *A term-indexed relation $\mathscr{R}$ is a family of binary relations between closed processes indexed by term environments. We write $\rho\models P\mathscr{R}Q$ (or equivalently $P\ \mathscr{R}_\rho\ Q$) to mean that P and Q are related by $\mathscr{R}$ at ρ and that $\mathit{fn}(P,Q)\subseteq\mathit{fn}(\rho)$.*

We have a notion of contextuality corresponding to that given in Def. 2. We note $\rho\setminus n$ the term environment resulting from erasing all bindings M/x such that $n\in\mathit{fn}(M)$.

Definition 6 (Contextuality). *A term-indexed relation $\mathscr{R}$ is contextual whenever $\rho\models P\mathscr{R}Q$ implies (i) if $\rho\vdash R$ then $\rho\models P\,|\,R\rho\mathscr{R}Q\,|\,R\rho$, (ii) $\rho,n/x\models P\mathscr{R}Q$ with $n\notin\mathit{fn}(\rho)$, and (iii) $\rho\setminus n\models(\mathsf{new}\ n)P\mathscr{R}(\mathsf{new}\ n)Q$.*

The barb predicate is defined relative to a term-environment, as expected: $\rho\models P\downarrow_a \triangleq a\in\mathscr{A}(\rho)\wedge P\downarrow_a$, and $\rho\models P\Downarrow_a \triangleq a\in\mathscr{A}(\rho)\wedge P\Downarrow_a$.

Definition 7 (Behavioural equivalence). . *Behavioural equivalence, noted $\cong^{A\pi}$, is the largest symmetric and contextual term-indexed relation $\mathscr{R}$ such that $\rho\models P\mathscr{R}Q$ implies (i) if $\rho\models P\downarrow_n$ then $\rho\models Q\Downarrow_n$, and (ii) if $P\xrightarrow{\tau}P'$ then $\exists Q'$. $Q\Longrightarrow Q'$ and $\rho\models P'\mathscr{R}Q'$.*

4 The implementation

Our assumptions about the low-level communication model are the same as those of [3]. In particular, we presuppose a Dolev-Yao network model, in which an intruder can interpose a computer in all communication paths and thus alter

Table 2 Labelled transitions for the applied pi

$$\text{(OUT)}\quad \frac{}{a\langle \tilde{M}\rangle \xrightarrow{a\langle \tilde{M}\rangle} \mathbf{0}} \qquad \text{(IN)}\quad \frac{}{a(\tilde{x}).P \xrightarrow{a(\tilde{M})} P\{\tilde{M}/\tilde{x}\}}$$

$$\text{(OPEN)}\quad \frac{P \xrightarrow{(\tilde{b})\, a\langle \tilde{M}\rangle} P' \qquad c \neq a,\ c \in fn(\tilde{M})}{(\mathsf{new}\ c)P \xrightarrow{(\tilde{b},c)\ a\langle \tilde{M}\rangle} P'}$$

$$\text{(CLOSE)}\quad \frac{P \xrightarrow{(\tilde{b})\, a\langle \tilde{M}\rangle} P' \quad Q \xrightarrow{a(\tilde{M})} Q' \qquad \tilde{b} \notin fn(Q)}{P \mid Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{b})(P' \mid Q')}$$

$$\text{(LET)}\quad \frac{d(\tilde{M}) \to N \quad P\{N/x\} \xrightarrow{\alpha} P'}{\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \xrightarrow{\alpha} P'} \qquad \text{(LET-ELSE)}\quad \frac{d(\tilde{M}) \not\to \quad Q \xrightarrow{\alpha} Q'}{\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \xrightarrow{\alpha} Q'}$$

or copy parts of messages, replay messages or forge new ones. We also assume that each principal has a secure environment in which to compute and store private data. On the other hand, we assume that the intruder cannot gain control of the whole network, and thus we do not guarantee that it actually will intercept every message. Consequently, message delivery may always be achieved with an adequate degree of redundancy. All processes can send and receive messages through a network interface consisting of a channel *net*. Typically all the exchanges over this channel net are encrypted. As [3] our results about the implementation rely on the presence of noise to prevent traffic analysis. Given the simple network interface, injecting noise into the network is simply accomplished by the process

$$W \triangleq !(\mathsf{new}\ n)!net\langle\{n\}_n\rangle$$

which generates infinitely many copies of infinitely many secret packets.

4.1 Data structures for names

The one-way constructors $rd(n), wr(n)$ (read and write seeds associated to n) are used to form the encryption keys employed in the representation of names. Given a name n, its representation as a fully-fledged channel includes the name identity and two encryption keys corresponding to the read and write capabilities: $(pub(n), ek(wr(n)), ek(rd(n)))$. The effect of casting a name at a higher type (i.e. as a read/write only channel, or top) is realized by means of a mechanism that masks away some, or all, of the encryption keys corresponding to the high-level type capabilities. We discuss how this is accomplished below.

Names come always equipped with self-signed certificates of the form

$$Cert(n) \triangleq \{pub(n), hash(ek(wr(n))), hash(ek(rd(n)))\}_{priv(n)}$$

The certificates help determine the "type" of the names they are attached to. Suppose we receive a tuple formed as follows: $(M_0, M_1, M_2, Cert(n))$. We first ensure that the certificate corresponds to the correct identity by using the public key to verify the certificate and then match the first certified argument and the public key. Then one may decide whether or not M_1 and M_2 are valid keys for the identity M_0 by calculating $hash(M_1)$ and $hash(M_2)$ and by checking whether if $equals(hash(M_1), hash(ek(wr(n))))$ and $equals(hash(M_2), hash(ek(rd(n))))$ reduce: in the former case we certify a write capability, in the latter a read capability.

We henceforth let $\underline{n}$ note the representation of n together with the associated certificate:

$$\underline{n} \;=\; (pub(n), ek(wr(n)), ek(rd(n)), Cert(n))$$

and define the following naming convention for the components:

$$n_{ID} = \pi_1(\underline{n}),\ n_w^+ = \pi_2(\underline{n}),\ n_r^+ = \pi_3(\underline{n}),\ n_{CERT} = \pi_4(\underline{n})$$

A corresponding naming scheme applies to variables, namely: given a variable x, we write $\underline{x}$ for a tuple of variables[1] $(x_{ID}, x_w^+, x_r^+, x_{CERT})$, so that

$$x_{ID} = \pi_1(\underline{x}),\ x_w^+ = \pi_2(\underline{x}),\ x_r^+ = \pi_3(\underline{x}),\ x_{CERT} = \pi_4(\underline{x})$$

Based on these conventions, we introduce useful notation to express various operations to manipulate the representation of names. We let let $x = Typeof(\underline{u})$ in P note the process that verifies that $\underline{u}$ represents a name certified by u_{CERT}, determines the true type of $\underline{u}$, and bind that type to x; in such case we informally say that $\underline{u}$ has such type. The types from the high-level calculus are represented in the implementation in the simplest possible way, namely: by means of nullary constructors: $\mathsf{rw}, \mathsf{r}, \mathsf{w}$ for the corresponding channel types, and $\top$ is for the type $\top$. We then let let $x = Cast(\underline{u}, T)$ in P be a process that casts the type of $\underline{u}$ to T and binds the resulting term and the certificate to x in P. Finally, the process

[1] Notice the difference: in a term the subscript indicates the application of a constructor, in a variable it does not and it's simply used as an index.

let $x = Meet((\underline{u},\underline{v}))$ in P computes the meet of $\underline{u}$ and $\underline{v}$ whenever $u_{CERT} = v_{CERT}$: in that case, it constructs a new tuple representing the result of merging the capabilities in $\underline{u}$ and $\underline{v}$ binding it to x.

All these operations can be encoded with little effort by nested applications of projection and equality destructors: we omit all details. We remark that casts and meets always applied to certified name representations. In fact, our protocols satisfy the following invariants: (i) whenever we evaluate let $x = Cast(\underline{u}, T)$ in P, u_{CERT} is indeed a certificate that certifies the remaining components of $\underline{u}$, and the type of $\underline{u}$ is $S :> T$ (ii) whenever we evaluate let $x = Meet(\underline{u},\underline{v})$ in P, u_{CERT} and v_{CERT} are the same certificate which certifies the remaining components of $\underline{u}$ and $\underline{v}$ at their respective types.

4.2 Message Filtering

As in [3], our implementation relies on the ability of processes to filter replays of messages, based on nonces. We write if $M \notin Set_n$ then P, for a process that adds M to the set of messages on the channel n, and, in case M does not belong to the set, continues as P . We omit the rather obvious details of how the testing process if $M \notin N$ then P else Q can be implemented.

$$\text{if } M \notin Set_n \text{ then } P \triangleq n(y).(\text{if } M \notin y \text{ then } n\langle M :: y\rangle \,|\, P \text{ else } n\langle y\rangle$$

The output of a message M on channel c is realized by a simple protocol that emits (the translation of) M on the *net* encrypted with the write encryption key associated with c. To ensure delivery, the emission is replicated, and packaged with a fresh nonce to protect against replay attacks. The nonce also acts as a confounder for cryptanalysis attacks.

$$\mathsf{emit}(\{m\}_k) \;\triangleq\; (\text{new } n)!\, net\langle\{m,n\}_k\rangle$$

The input of a message relies on two filtering protocols. The first reads from the *net* and proceeds with the continuation if the message is successfully decrypted; otherwise it re-emits the message and retries:

$$\begin{array}{l}\text{filter } \tilde{y} \text{ with } N \text{ in } P \;\triangleq\\ \quad \text{rec}X.net(x).\text{decrypt } x \text{ as } \{\tilde{y}\}_N \text{ in } P \text{ else } (net\langle x\rangle \,|\, X)\end{array}$$

The second protocol filters messages based on their types. We write if $WF(\tilde{x},t)$ then P else Q for the process that tests whether $\tilde{x}$ is certified and has type s, for some $s <: t$.

$$\begin{array}{l}\text{filter } \tilde{y} \text{ from } c@t \text{ in } P \;\triangleq\; \text{rec}X.c(\tilde{x}).\\ \quad \text{if } WF(\tilde{x},t) \text{ then } (\text{let } \tilde{y} = Cast((\tilde{x},t)) \text{ in } P) \text{ else } (c\langle\tilde{x}\rangle \,|\, X)\end{array}$$

4.3 First Translation

As the first step we give the representation of names, which arises as expected:

$$\begin{array}{lcl}[\![u@\mathsf{rw}]\!] & \triangleq & (u_{ID}, u_w^+, u_r^+, u_{CERT})\\ [\![u@\mathsf{r}]\!] & \triangleq & (u_{ID}, hash(u_w^+), u_r^+, u_{CERT})\\ [\![u@\mathsf{w}]\!] & \triangleq & (u_{ID}, u_w^+, hash(u_r^+), u_{CERT})\\ [\![u@\top]\!] & = & (u_{ID}, hash(u_w^+), hash(u_r^+), u_{CERT})\end{array}$$

Next, we detail the protocols for input/output.

Write protocol. On the client side, writing on channel is accomplished by emitting a packet encrypted under the channel's write encryption key. The message is replicated to ensure that it is eventually delivered, and packaged with a nonce to protect against replay attacks. The server, in turn, uses the write decryption key to receive the message and then stores it in a private queue. The server uses the nonce to filter multiple copies of the message and stores them into a private queue n°. It also filters based on the format of the messages received, requiring that they match the format expected of the encoding of names.

Read Protocol. On the client side, the reader process uses the channel's read encryption key to send a read request to the channel server: the request takes the form of an encryption key that will be used to exchange the message with the server. The client then waits for a message from the server encrypted under the session key: upon receiving the packet, it proceeds with its continuation. The server, in turn, uses the channel's read decryption key to receive a request from the client. Each request is packaged with the channel's read encryption key and comes as a triple that includes an encryption key, the representation of a type together with a nonce: the server uses the type to select one of the message from its private queue and then packages the messages with the key. To protect against replays, the server keeps track of the nonces received on a private channel n^* (hardly realistic, of course, indeed, the solution from [3], based on challenge response mechanisms can be employed here). The nonce can be spared on the actual message sent by the server as the key expires at the completion of the protocol (the server may easily filter out replays of the session key).

The definitions, in Table 3, formalize the intuitions given above, and are applied implicitly only on well-typed processes. We use the notation n_r^- and n_w^- to refer to $dk(rd(n))$ and $dk(wr(n))$ respectively. The encoding of a restriction generates a corresponding name and its associated channel. As for matching, a name-equality test in the source calculus is implemented as a corresponding test on the public keys associated with the pi names. In addition, in case the test is successful, the continuation process $[\![P]\!]$ is given access to the new set of capabilities that corresponds to the meet of the two types associated with the pi names in the typing

derivation of the pi process. Notice that the meet process cannot be stuck, since (i) representations $\underline{u},\underline{v}$ obtained from the translation are well-formed and (ii) representations $\underline{u},\underline{v}$ obtained from the channels have been erased from potential malformed terms, i.e. there are types T,T' such that $\underline{u} = [\![u@T]\!], \underline{v} = [\![v@T']\!]$.

4.4 Soundness

The synchronization two protocols ensure the following properties: (i) each message output by a writer will reach at most one reader, and dually, (ii) a legitimate reader client will complete the protocol provided that a type-compatible message on the same channel has been output by a writer. Thus, if we allocate channels for the free names of the high-level processes, we can prove that any pi synchronization on a name is simulated by a corresponding reduction in the implementation, and conversely that the synchronizations on the channels queue of the implementation reflect the τ reductions of the source calculus.

Given a type environment I, we define the term environment corresponding to I, noted $\{\!|\,\mathrm{I}\,|\!\}$:

$$\{\!|\,\emptyset\,|\!\} = \{net/x_o\}, \quad \{\!|\,\mathrm{I}, a:A\,|\!\} = \{\!|\,\mathrm{I}\,|\!\}, [\![a:A]\!]/\underline{x}$$

where $\underline{x} \notin \mathrm{dom}(\{\!|\,\mathrm{I}\,|\!\})$. Then we define the computing environment which includes the interface to the network, the noise-generating process and channel support for the free names shared between the processes and the environment.

$$\mathsf{E}_{\mathrm{I}}[-] \;=\; -\,|\; W \;|\; \prod_{n\in \mathrm{dom}(\mathrm{I})} \mathit{Chan}_n$$

We have finally all we need to establish the result of operational correspondence.

Theorem 1 (Operational Correspondence).

- *If* $P \xrightarrow{\tau} P'$ *then* $\mathsf{E}_{\mathrm{I}}[[\![P]\!]] \Longrightarrow \approx^{A}_{\{\!|\,\mathrm{I}\,|\!\}} \mathsf{E}_{\mathrm{I}}[[\![P']\!]]$.
- *Conversely, let* $H \approx^{A}_{\{\!|\,\mathrm{I}\,|\!\}} \mathsf{E}_{\mathrm{I}}[[\![P]\!]]$*; if* $H \xrightarrow{\tau} K$ *then there exists* P' *s.t.* $P \Longrightarrow P'$ *and* $K \approx^{A}_{\{\!|\,\mathrm{I}\,|\!\}} \mathsf{E}_{\mathrm{I}}[[\![P']\!]]$.

The preservation direction of this result is standard. On the other hand, 'the 'reflection" direction is subtle, as the translation is not "prompt" [12]: in fact, it takes several steps for $\mathsf{E}_{\mathrm{I}}[[\![P]\!]]$ to be ready for the commit synchronization step on the channel queue that corresponds to the high-level synchronization on the channel. As it turns out, however, these steps are not observable and can be factored out in the proof by resorting to a suitable notion of *administrative* equivalence (noted $\approx^{A}_{\rho}$ in the statement of Theorem 1) included in $\cong^{A\pi}_{\rho}$ (see the Appendix for details). Because of this inclusion, the soundness of the translation is a a direct consequence of Theorem 1.

Theorem 2. *If* $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{E}_{\mathrm{I}}[[\![P]\!]] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[[\![Q]\!]]$*, then* $\mathrm{I} \models P \cong^{\pi} Q$.

The converse direction of Theorem 22 does not hold. In fact, as we noted, the communication protocols presuppose a certain structure associated with names. Indeed, for the names that are statically shared with the context, this structure is easily enforced by allocating the corresponding channels as part of the encoding definition. However, the context may dynamically generate new names that do not satisfy the expected invariants. Notice, for instance, that the client of a reader protocol presupposes a legitimate channel on the other end of the protocol and is not protected against malformed messages received by illegitimate channels: given that, it is easy to find a counter-example to full abstraction. For instance, in the source calculus we have:

$$a : \mathsf{w} \models a(y@\mathsf{rw}).y(x@\mathsf{rw}).y\langle x@\mathsf{rw}\rangle \cong^{\pi} a(y@\mathsf{rw})$$

This is an instance of the well-known asynchronous pi calculus law $a(x).a\langle x\rangle \cong \mathbf{0}$, and holds in our pi calculus for similar reasons. On the other hand, one easily sees that

$$a^{+}_{w}/z \not\models \mathsf{E}_{\mathrm{I}}[[\![a(y@\mathsf{rw}).y(x@\mathsf{rw}).y\langle x@\mathsf{rw}\rangle]\!]] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[[\![a(y@\mathsf{rw})]\!]]$$

In fact, a context may create the legitimate representation of a full-fledged name $\underline{b}$ and exchange it over a; the subsequent request $\mathsf{emit}(\{sk(k),\mathsf{rw}\}_{b^{+}_{r}})$ made by the left process can now be decrypted by the context, which possesses the decryption key b^{-}_{r}, and thus the left reduct $[\![b(x@\mathsf{rw}).b\langle x@\mathsf{rw}\rangle]\!]$ can be distinguished from the null process.

4.5 A Fully Abstract Translation

To recover full abstraction, we must shield our translated processes from such undesired interactions. That may be achieved by setting up the synchronization protocols so as to ensure that all the exchanges occur over system-generated, trusted channels whose decryption keys remain secret.

The new translation introduces a separation between *client names*, used syntactically by context processes and by translated processes to communicate, and corresponding *server names* generated within the system and associated with system generated channels to be employed in the actual protocols for communication.

A proxy server maintains an association map between client and server names so as to preserve the expected interactions among clients. The map is implemented as a set of entries of the form $(\mathit{pub}(n),\underline{m})$, whose intended invariant is that m is the server counterpart of the client name n. We call $\mathit{pub}(n)$ the index of the entry, and $\underline{m}$ the target The proxy map is set up to ensure that each index has exactly one target. We represent the public keys used by the proxy service by letting

Table 3 First Translation

Channels

$$
\begin{array}{rcl}
WS_n & \triangleq & !\,\mathsf{filter}\ (\underline{x},z)\ \mathsf{with}\ n_w^-\ \mathsf{in\ if}\ z \notin Set_{n^*}\ \mathsf{then}\ n^\circ\langle\underline{x}\rangle \\
RS_n & \triangleq & !\,\mathsf{filter}\ (y,t,z)\ \mathsf{with}\ n_r^-\ \mathsf{in\ if}\ z \notin Set_{n^*}\ \mathsf{then\ filter}\ \underline{x}\ \mathsf{from}\ n^\circ@t\ \mathsf{in}\ !\,net\langle\{\underline{x}\}_y\rangle \\
Chan_n & \triangleq & (\mathsf{new}\ n^*,n^\circ)\ n^*\langle\emptyset\rangle \,|\, RS_n \,|\, WS_n
\end{array}
$$

Clients – The clauses for composition and replication are defined homomorphically: $[\![P\,|\,Q]\!] = [\![P]\!]\,|\,[\![Q]\!]$, $[\![!\,P]\!] = !\,[\![P]\!]$.

$$
\begin{array}{rcl}
[\![u\langle v@T\rangle]\!] & \triangleq & \mathsf{emit}(\{[\![v@T]\!]\}_{u_w^+}) \\
[\![u(x@T).P]\!] & \triangleq & (\mathsf{new}\ k)\,\mathsf{emit}(\{sk(k),T\}_{u_r^+})\,|\,\mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ [\![P]\!] \\
[\![(\mathsf{new}\ n)P]\!] & \triangleq & (\mathsf{new}\ n)(Chan_n\,|\,[\![P]\!]) \\
[\![[u=v]P;Q]\!] & \triangleq & \mathsf{if}\ u_{ID}=v_{ID}\ \mathsf{then}\ (\mathsf{let}\ \underline{t}=Meet(\underline{u},\underline{v})\ \mathsf{in}\ [\![P\{t/u,t/v\}]\!])\ \mathsf{else}\ [\![Q]\!] \qquad t \notin fn(P)
\end{array}
$$

$k_P^+ \triangleq ek(k)$ and $k_p^- \triangleq dk(k)$, with k a seed not known to the environment.

The read/write protocols follow the same rationale as in the previous translation, with the difference that now the clients must first obtain the access to the system channel by contacting the proxy server. The interaction between clients and proxy is as follows: the client presents a name to the proxy and the proxy replies with the corresponding server name cast at the (true) type of the name sent by the client. In case the name received is new, the proxy returns a fresh server name for which it also allocates a system channel. On the client side, the protocol is implemented as shown below:

$$
\begin{array}{l}
\mathsf{link}\ (\underline{u},\underline{y})\ \mathsf{in}\ P \triangleq \\
\quad (\mathsf{new}\ h)\mathsf{emit}(\{sk(h),\underline{u},\}_{k_p^+})\,|\,\mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(h)\ \mathsf{in}\ P
\end{array}
$$

For the proxy side, the definition is found in Table 4. The only subtlety is that upon receiving a name that does not occur in the association map, the proxy allocates two indexes for the same target: one index is the name received from the client, the other is the public key of the target itself. This second association is needed to make linking idempotent, so that linking a server name always returns the same name. One may wonder how a client could possibly end up requesting a link for a server name as (i) server names originate from the proxy, and (ii) are never passed on any exchange by the clients. Notice however, that this invariant is only true of the clients that arise from the translation, not for arbitrary applied pi processes of the context.

Having given the intuitions, the definitions in Table 4 should be easily understood. Notice that the clause for restriction is defined homomorphically in this translation, as the creation of the channel is delegated entirely to the proxy server. In the translation of the matching construct, we could test matching on linked names (which would make the translation more uniform), rather than on client names. While this choice has no consequences in the centralized translation, it does create a problem in the distributed implementation (see Section 4.6: in that case, matching should test names linked on the same proxy and this creates a technical problem as names are known at different proxies).
We write $\mathsf{let}\ \tilde{y} = ?(x_{ID},z)\ \mathsf{in}\ P$ for the process that extracts the target $\tilde{y}$ associated to x_{ID} in z and continues as P.

Full Abstraction Having set up the underlying infrastructure, we now have the expected protection against hostile contexts. We may therefore strengthen the result of Theorem 22 as desired, provided that we plug our processes in the appropriate computing environment. We first define

$$\mathsf{CE}[-] \;=\; -\,|\,W\,|\,Proxy$$

and let the low-level term environments corresponding to the high-level type environment be extended with a new binding expressing the knowledge of the public proxy encryption key k_P^+ needed to interact with the proxy. Then we have:

Theorem 3. $\mathrm{I} \models P \cong^{\pi} Q \iff \{\!|\,\mathrm{I}\,|\!\}, k_P^+/y \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$.

4.6 A Distributed Implementation

While the use of the proxy server to protect against misbehaved channels is effective in achieving full abstraction, it is clear that a centralized implementation as the one we just described in hardly realistic. In this section, we discuss a new, distributed implementation that distributes the proxy services among different servers. The new solution is based is based on the idea of partitioning the network in domains each of which administrated by a proxy server.

To model this partitioning of the network, we extend our high level calculus with the syntactic category of nets, which are simply processes labelled with a domain label.

Table 4 Fully Abstract Translation

Proxy server

$$\begin{array}{rcl} P_t & \triangleq & !\,\mathsf{filter}\ (k,\underline{x},y)\ \mathsf{with}\ k_p^-\ \mathsf{in}\ \mathsf{if}\ y \notin Set_{t^*}\ \mathsf{then}\ \mathsf{let}\ s = Typeof(\underline{x})\ \mathsf{in} \\ & & \quad t(z).\mathsf{let}\ \tilde{y} = ?(x_{ID},z)\ \mathsf{in}\ t\langle z\rangle \mid \mathsf{let}\ \tilde{z} = Cast(\tilde{y},s)\ \mathsf{in}\ !net\langle\{\tilde{z}\}_k\rangle \\ & & \quad\quad \mathsf{else}\ (\mathsf{new}\ n)Chan_n \mid\ t\langle z :: (x_{ID};\underline{n}) :: (n_{ID};\underline{n})\rangle\ \mid \mathsf{let}\ \tilde{z} = Cast(\underline{n},s)\ \mathsf{in}\ !net\langle\{\tilde{z}\}_k\rangle \\ Proxy & \triangleq & (\mathsf{new}\ t,t^*)\ P_t \mid t\langle\emptyset\rangle \mid t^*\langle\emptyset\rangle \end{array}$$

Channels – The definition of $Chan_n$ is unchanged from Table 3

Clients – The clauses for composition and replication are defined homomorphically: $\langle\!\langle P \mid Q\rangle\!\rangle = \langle\!\langle P\rangle\!\rangle \mid \langle\!\langle Q\rangle\!\rangle$, $\langle\!\langle !\,P\rangle\!\rangle = !\,\langle\!\langle P\rangle\!\rangle$.

$$\begin{array}{rcl} \langle\!\langle u\langle v@T\rangle\rangle\!\rangle & = & \mathsf{link}\ (\underline{u},\underline{x})\ \mathsf{in}\ \mathsf{emit}(\{\![v@T]\!\}_{x_{\mathsf{w}}^+}) \\ \langle\!\langle u(x@T).P\rangle\!\rangle & = & \mathsf{link}\ (\underline{u},\underline{y})\ \mathsf{in}\ (\mathsf{new}\ k)\,\mathsf{emit}(\{sk(k),T\}_{y_{\mathsf{r}}^+}) \mid \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\!\langle P\rangle\!\rangle \\ \langle\!\langle [u=v]P;Q\rangle\!\rangle & = & \mathsf{if}\ u_{ID} = v_{ID}\ \mathsf{then}\ (\mathsf{let}\ \underline{t} = Meet((\underline{u},\underline{v})\ \mathsf{in}\ \langle\!\langle P\{t/u,t/v\}\rangle\!\rangle)\ \mathsf{else}\ \langle\!\langle Q\rangle\!\rangle \qquad t \notin fn(P) \\ \langle\!\langle (\mathsf{new}\ n)P\rangle\!\rangle & = & (\mathsf{new}\ n)\langle\!\langle P\rangle\!\rangle \end{array}$$

Nets are composed according to the following syntax:

$$S,T\ ::=\ \delta\{P\}\ \mid\ S \mid T\ \mid\ (\mathsf{new}\ n)S\ \mid\ \mathbf{stop}$$

Domain names, ranged over by δ are drawn from a denumerable set of labels, disjoint from the set of names and the set of variables. We let $fd(S)$ be the set of domain labels in S. We emphasize that labels are not names, and are never exchanged over channels. The typing and dynamics of nets arise in the simples possible way from the corresponding notions defined for processes:

$$\text{(TYPING)}\quad \frac{\Gamma \vdash P}{\Gamma \vdash \delta\{P\}} \qquad\qquad \text{(DYNAMICS)}\quad \frac{P \xrightarrow{\alpha} Q}{\delta\{P\} \xrightarrow{\alpha} \delta\{Q\}}$$

As a result, domains have no impact on the dynamics and/or the typing of the high-level calculus: indeed, they serve a different purpose, namely to help devise the association of processes to proxies in the implementation. Notice, in particular, that the same (channel) name may be known at different domains: in the implementation, this will correspond to the name being represented by different channels, located at the different domains at which the name is known. Also, notice that the same domain may have multiple occurrences, as in $\delta\{P_1\} \mid \delta\{P_2\}$: based on the notion of dynamics introduced above, this net may be rewritten equivalently as $\delta\{P_1 \mid P_2\}$. Similarly, $(\mathsf{new}\ n)\delta\{P\}$ is the same as $\delta\{(\mathsf{new}\ n)P\}$.

Observational Equivalence The definition of observational equivalence for nets is inherited from that of processes. There is an important difference, however, in the notion of contextuality, in that a context may not include new domains, but only processes belonging to existing domains. This is ensured by the side condition $fd(U) \subseteq fd(S,T)$ definition below, and constitutes the key assumption for our distributed implementation, namely, we do not trust domains and proxy servers generated by the environment. While this is a somewhat strong assumption, on the other hand it appears to be realistic: notice, in fact, that the procedure of adding a domain to a network in real world scenarios requires physical authentication, rather than network protocols.

Definition 8 (Contextuality for Nets). *A type-indexed relation $\mathscr{R}$ over nets is contextual whenever*

- $\mathrm{I} \vDash S\mathscr{R}T$ *implies* $\mathrm{I}, a : A \vDash S\mathscr{R}T$ *and* $\mathrm{I} \vDash S \mid U\mathscr{R}T \mid U$ *for all U such that* $\mathrm{I} \vdash U$ *and* $fd(U) \subseteq fd(S,T)$*;*
- $\mathrm{I}, a : A \vDash S\mathscr{R}T$ *implies* $\mathrm{I} \vDash (\mathsf{new}\ a)S\mathscr{R}(\mathsf{new}\ a)T$

The definition of behavioural equivalence for nets, noted again $\cong^{\pi}$, arises now as expected, as the largest type-indexed equivalence relation which is contextual (in the sense above), barb preserving and reduction closed.

The new implementation Each domain in the high-level specification corresponds to a domain manager, which provides the proxy service to clients and manages the channels it has created; processes within a domain, in turn, are instructed to send their requests to the proxy associated with their enclosing domain. Since different proxies may have different entries for the same client name (remember that a name is possibly known in more domains), in the distributed implementation more channels servers may correspond to a pi calculus name. The domain managers must therefore provide a further service, to manage the queues located at the distributed channels associated to the same client name. This, in turn, is based on further, domain, service to gain access to fellows proxies.

Table 5 Distributed Translation

Proxy Service

$$\begin{aligned} P^{\delta}_{q,t} \triangleq\ & !\,\mathsf{filter}\ (k,\underline{x},y)\ \mathsf{with}\ \delta^{-}_{p}\ \mathsf{in\ if}\ y \notin Set_{t^*}\ \mathsf{then\ let}\ s = Typeof(\underline{x})\ \mathsf{in} \\ & t(z).\mathsf{let}\ \tilde{y} = ?(x_{ID},y)\ \mathsf{in}\ t\langle z\rangle \,|\, \mathsf{let}\ \tilde{z} = Cast(\tilde{y},s)\ \mathsf{in}\ !net\langle\{\tilde{z}\}_k\rangle \\ & \quad \mathsf{else}\ (\mathsf{new}\ n)Chan_n[q(X).(q\langle X\rangle \,|\, \Pi_{k_\delta \in X}\, !\,\mathsf{filter}\ \underline{w}\ \mathsf{from}\ n^{\circ}@\top\ \mathsf{in}\ \mathsf{emit}(\{x_{ID},\underline{w}\}_{k_\delta}))] \\ & \quad\ \mid t\langle z :: (x_{ID};\underline{n}) :: (n_{ID};\underline{n})\rangle \ \mid \mathsf{let}\ \tilde{z} = Cast(\underline{n},s)\ \mathsf{in}\ !net\langle\{\tilde{z}\}_k\rangle \end{aligned}$$

Queue Service

$$\begin{aligned} Q^{\delta}_{q,t} \triangleq\ & !\,\mathsf{filter}\ (x,\underline{s},y)\ \mathsf{with}\ \delta^{-}_{q}\ \mathsf{in\ if}\ y \notin Set_{q^*}\ \mathsf{then}\ t(z).(t\langle z\rangle \,| \\ & \quad \mathsf{let}\ \tilde{y} = ?(x,z)\ \mathsf{in}\ \mathsf{emit}(\{\underline{s}\}_{y^{+}_{w}})\ \mathsf{else} \\ & \quad\ (\mathsf{new}\ n)n\langle\rangle \,|\, q(X).(q\langle X\rangle \,|\, \Pi_{k_\delta \in X} n().\mathsf{emit}(\{x,\underline{s}\}_{k_\delta}))\,) \end{aligned}$$

Domain Service

$$D^{\delta}_{q,l} \triangleq \mathsf{emit}(\{\delta^{+}_{q}\}_{sk(k_D)}) \,|\, !\,\mathsf{filter}\ x,c\ \mathsf{with}\ sk(k_D)\ \mathsf{in\ if}\ c \notin Set_{l^*}\ \mathsf{then}\ q(y).q\langle y.x\rangle$$

Proxy Manager

$$M^{\delta} \triangleq (\mathsf{new}\ t,t^*,q,q^*,l^*)\ P^{\delta}_{q,t} \,|\, Q^{\delta}_{q,t} \,|\, D^{\delta}_{q,l} \,|\, t\langle\emptyset\rangle \,|\, t^*\langle\emptyset\rangle \,|\, q\langle\emptyset\rangle \,|\, q^*\langle\emptyset\rangle \,|\, l^*\langle\emptyset\rangle$$

Translation of nets: – the clauses for parallel composition and restriction are homomorphic

$$(\!|\delta\{P\}|\!) = M^{\delta} \,|\, (\!|P|\!)_{\delta}$$

Channels – The definition of $Chan_n$ is unchanged from Table 3

Clients – The clauses for matching, new, composition, replication are unchanged from Tab. 4

$$\begin{aligned} (\!|u\langle v@T\rangle|\!)_{\delta} &= \mathsf{link}^{\delta}\ (\underline{u},\underline{x})\ \mathsf{in}\ \mathsf{emit}(\{[\![v@T]\!]\}_{x^{+}_{w}}) \\ (\!|u(x@T).P|\!)_{\delta} &= \mathsf{link}^{\delta}\ (\underline{u},\underline{y})\ \mathsf{in}\ (\mathsf{new}\ k)\,\mathsf{emit}(\{k,T\}_{y^{+}_{r}}) \,|\, \mathsf{filter}\ \underline{x}\ \mathsf{with}\ k\ \mathsf{in}\ (\!|P|\!)_{\delta} \end{aligned}$$

We represent public keys used by the proxy service by using the one-way function $p(x)$ and by letting $\delta^{+}_{p} \triangleq ek(p(\delta))$ and $\delta^{-}_{p} \triangleq dk(p(n))$; these keys corresponds to the keys k^{+}_{P}, k^{-}_{P} utilized in the centralized translation. We use an one-way function $q(x)$ to represent the public keys used by the queue service: $\delta^{+}_{q} \triangleq ek(q(\delta))$ and $\delta^{-}_{q} \triangleq dk(q(\delta))$. We assume that domain managers are connected via secure links, represented by a shared key $sk(k_D)$ generated from the private name k_D. Finally, we introduce the following two bits of new notation. $Chan[P]$ indicates the process $(\mathsf{new}\ n^*,n^{\circ})\ n^*\langle\emptyset\rangle \,|\, RS_n \,|\, WS_n \,|\, P$. The definition of links is extended as expected, with a parameter corresponding to the assigned proxy:

$$\begin{aligned} \mathsf{link}^{\delta}\ (\underline{u},\underline{y})\ \mathsf{in}\ P \triangleq\ & (\mathsf{new}\ k)\mathsf{emit}(\{sk(k),\underline{u},\}_{\delta^{+}_{p}}) \\ & \mid \mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P \end{aligned}$$

The new implementation is given in Table 5. Each proxy has three threads $P^{\delta}_{q,t}$, $Q^{\delta}_{q,t}$, $D^{\delta}_{q,l}$ responsible for the proxy, queue and domain services, respectively. The three threads share a channel q collecting the public keys giving access to fellow proxies. The domain service $D^{\delta}_{q,l}$ is responsible for updating this queue and for publishing the public queue key of the proxy associate to the domain δ.

The proxy and the queue service also share the table of binding client and server names stored on the private channel t. The new definition of the proxy service extends the one given in Table 4, with the addition of a replicated read request into the new channel created when the client name is not present in the proxy's table. The read request is composed by several requests which repeatedly extract messages from the channel queue channel; such requests are directed to all the domains known at the time the channel was created. The extracted message and the client name index associated to the channel are then safely sent to the given queue managers encrypted under their public queue key. We call the replicated read request $!\,\mathsf{filter}\ \underline{w}\ \mathsf{from}\ n^{\circ}@\top\ \mathsf{in}\ \mathsf{emit}(\{M,\underline{w}\}_{\delta^{+}_{q}})$ a *forwarder* to the domain δ. The queue service $Q^{\delta}_{q,t}$ waits for the packets sent by the forwarders and by other domain managers. The server retrieves the name index and checks if the entry is associated to a channel. If it is, the message is sent to the queue of the associated channel. Otherwise both

the name index and the message are non-deterministically sent to a manager of some known domain.

The translation of nets is compositional, and does not rely on any pre-existing infrastructure for communications, as now the proxies are dynamically generated within the translation; as in previous implementations we assume the presence of noise on the communication interface. We extend the low-level term environment corresponding to the high-level type environment with the public encryption keys of the proxy managers corresponding to the high-level free domains: $\{\!|\delta_1,\ldots,\delta_n|\!\} \triangleq \{\delta_1{}_p^+/x,\ldots,\delta_n{}_p^+/z\}$. We finally obtain:

Theorem 4. *$\mathrm{I} \models S \cong^{\pi} T$ if and only if $\{\!|\mathrm{I}|\!\}, \{\!|fd(S,T)|\!\} \models W \mid (\mathsf{new}\ k_D)(\!(S)\!) \cong^{A\pi} W \mid (\mathsf{new}\ k_D)(\!(T)\!)$.*

5 Conclusion

We have developed a secure implementation of a typed pi calculus, in which the access to communication channels is regulated by capability types. The implementation draws on a representation of the typed capabilities in the high-level calculus as term capabilities protected by encryption keys only known to the intended receivers. The implementation relies on a proxy service to protect against malformed messages from the environment. This is achieved by generating certified names (and associated channels) to represent the context-generated names within the system. We have also developed a distributed implementation in which the certification service is implemented by a set of distributed proxies. Being fully compositional, the distributed implementation appears to be adequate for open-ended networks. The only limitation in this respect is represented by our current assumption that all proxies that participate in the synchronization protocols be fully trusted. While a certain degree of trust appears necessary to achieve a secure implementation, it would be desirable to have some form of guarantees also in the presence of malicious proxies. Achieving that seems feasible with our implementation, by strengthening the protocols that govern the interactions among proxies. We leave this to our plans of future work.

Our translation has several analogies with previous attempts in the literature [3, 9]. In [3], the authors provide a fully abstract implementation of the join calculus into a dialect of the calculus equipped with cryptographic primitives. The located nature of channels in the join calculus makes it possible to rely on a very compact representation in which a channel is associated a with a low level key name a^+, and to protect communications using an asymmetric cryptosystem. In [9] a fully abstract implementation of the pi calculus without matching into the join calculus is given; similarly to ours, their translation relies both on the presence of proxy pairs of internal and external names, and on relays among the pairs components. By composing the encodings [9],[3] one can securely implement the untyped pi calculus without matching in a join calculus equipped with cryptographic primitives. However, it is not clear how to implement matching following this approach. In fact, as noted in [15], the ability of test syntactic equality on names invalidates the semantic equalities on names provided by the equators [11] used in [9] to merge internal and external names.

Acknowledgments. The name representation based on self-signed certificates was suggested to us by Cedric Fournet. We would like to thank him for his insightful comments and constructive criticism on a previous draft of the paper.

References

[1] M. Abadi. Protection in programming-language translations. In *Proc. of ICALP '98*, pages 868–883, 1998.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL '01)*, pages 104–115. ACM Press, 2001.

[3] M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. *Information and Computation*, 174(1):37–83, April 2002.

[4] M. Baldamus, J. Parrow, and B. Victor. A fully abstract encoding of the π-calculus with data terms. In *Proc. of ICALP '05*, pages 1202–1213, 2005.

[5] B. Blanchet. From Secrecy to Authenticity in Security Protocols. In M. Hermenegildo and G. Puebla, editors, *9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *Lecture Notes on Computer Science*, pages 342–359, Madrid, Spain, Sept. 2002. Springer Verlag.

[6] B. Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, Oakland, California, May 2004.

[7] M. Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theor. Comput. Sci.*, 195(2):205–226, 1998.

[8] M. Bugliesi and M. Giunti. Typed processes in untyped contexts. In R. Nicola and D. Sangiorgi, editors, *Proc. of TGC 2005, Symposium on Trustworthy Global Computing*, volume 3705 of *Lecture Notes on Computer Science*, pages 19–32. Springer-Verlag, 2005.

[9] C. Fournet. *The Join-Calculus: a Calculus for Distributed Mobile Programming*. PhD thesis, Ecole Polytechnique, Palaiseau., November 1998. Also published by INRIA, TU-0556.

[10] M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2003.

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

[12] U. Nestmann and B. C. Pierce. Decoding choice encodings. *Inf. Comput.*, 163(1):1–59, 2000.

[13] B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5), 1996.

[14] B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.

[15] D. Sangiorgi and D. Walker. *The π-calculus A theory of mobile processes*. Cambridge, 2001.

A Proofs

The appendix is structured as follows. We first define administrative reductions and equivalence and we show that administrative reductions are closed under administrative equivalence and that administrative equivalence is contained in behavioural equivalence. Next we state a number of lemmas which ensure that the leftover of the communications of the protocol are invisible to the context (cf [3]). Subsequently we sketch the proofs of the soundness and completeness of the centralized encoding $\langle\!\langle \cdot \rangle\!\rangle$. We omit the proofs of full abstraction of the distributed encoding which follow the same rationale.

Administrative reductions and equivalence As we introduced in §4, our encodings are not prompt [12], i.e. $\mathsf{CE}[\langle\!\langle P \rangle\!\rangle]$ does several steps to be ready to execute the commit synchronization step on the channel queue that corresponds to the high-level synchronization on the channel. Such reductions have the following properties: each reduction conducing to a commit (i) does not preclude any other reduction and (ii) is invisible to the context. As we will show later, these reductions preserve behavioural equivalence.

Based on these intuitions, in the following we introduce the notion of administrative reduction. A reduction is administrative whenever satisfies one of the clauses of the Definition 9 introduced some lines below; we assume that the processes of interest use a public interface for communications. The first clause is clear. The second clause describes a protocol that uses a symmetric cryptoscheme and that is formed by a recursive filter meta-process waiting for packets on the public interface and a replicated public emission of the packet waited by the filter; the reduction involving the filter receiving the packet is administrative whenever the content of the packet has the correct arity and there not exists another packet with different content that could be accepted by the filter.

The third clause describes a similar protocol involving a recursive filter and a public output and says that a reduction is administrative whenever is inferred from the filter receiving a wrong packet (i.e., encrypted with a different key or containing terms mismatching the expected arity). The fourth clause describes the behaviour of a filter of type T on a private channel n° which receives a term which represents a type S that is not a sub-type of T. The fifth clause describes the behaviour of a process which receives a set carried on by a private output $n^*\langle M \rangle$ and then tests if a term N belong to the set and if yes continue as Q; the reduction is administrative whenever the existence in the scope of n^* of a testing process on the same term N and continuing as Q' implies that $Q \equiv Q'$.

The sixth clause says that the deterministic reduction which "unblocks" $\mathsf{rec}\,X.P$ is administrative. The last clause describes a protocol where a table set carried on a private channel is managed in a way similar to that of our Proxy server (see Tab. 4). The managing of the table is done by a meta-process $T_N(Q)$ defined around Q; here Q is a process with free variables that will be closed in $T_N(Q)$. The reduction under analysis is the one where $T_N(Q)$ receives a table set from a private output $t\langle M \rangle$. Next the continuation of $T_N(Q)$ tests if M contains an index for N_{ID} (and in case binding the associated entry to the free variables of Q): if yes the process continue as $t\langle M \rangle \,|\, Q$ otherwise it creates a channel for the fresh name n, outputs on t the updated $M :: (N_{ID}; \underline{n}) :: (n_{ID}; \underline{n})$ and executes Q with $\underline{n}$ binding the free variables of Q.

In the following, for terms M, N we write $M = N$ to indicate that M is syntactically equal to N; we write $|M|$ to refer to the dimension of M.

Definition 9 (Administrative reduction). *Let $net \in fn(P)$ and suppose $P \xrightarrow{\tau} P'$. We say that $P \xrightarrow{\tau} P'$ is an administrative reduction, noted $P \xrightarrow{A} P'$, when $P\downarrow_a \Leftrightarrow P'\downarrow_a$ and one of the following cases holds:*

1. *the reduction is inferred from a synchronization among a replicated input on net and a replicated output on net;*
2. $P \equiv C[\mathsf{filter}_1\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P \,|\, !\,net\langle\{M\}_{sk(N)}\rangle]$ *and* $P' \equiv C[\mathsf{filter}_{2;\{M\}_{sk(N)}}\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P \,|\, !\,net\langle\{\{M\}_{sk(N)}\rangle]$ *and* $|M| = |\tilde{y}|$ *and* $\forall C', M' \,:\, P \equiv C'[net\langle\{M'\}_{sk(N)}\rangle] \wedge |M'| = |\tilde{y}| \quad . \quad M = M'$
   where $\mathsf{filter}_1\ \tilde{x}\ \mathsf{with}\ N\ \mathsf{in}\ P$ *is the process reached after unblocking the recursion guard of* $\mathsf{filter}\ \tilde{x}\ \mathsf{with}\ N\ \mathsf{in}\ P$ *by means of a deterministic reduction and* $\mathsf{filter}_{2;M}\ \tilde{x}\ \mathsf{with}\ N\ \mathsf{in}\ P$ *is the process Q s.t.* $\mathsf{filter}_1\ \tilde{x}\ \mathsf{with}\ N\ \mathsf{in}\ P \xrightarrow{c(M)} Q$;
3. $P \equiv C[\mathsf{filter}_1\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P \,|\, net\langle M\rangle]$ *and* $P' \equiv C[\mathsf{filter}_{2;M}\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P]$ *and* $\neg(M = \{M'\}_{sk(N)} \wedge |M'| = |\tilde{y}|)$;
4. $P \equiv C[(\mathsf{new}\ n^\circ)Q \,|\, \mathsf{filter}_1\ \tilde{y}\ \mathsf{from}\ n^\circ@T\ \mathsf{in}\ P \,|\, n^\circ\langle M\rangle]$ *and* $P' \equiv C[(\mathsf{new}\ n^\circ)Q \,|\, \mathsf{filter}_{2;M}\ \tilde{y}\ \mathsf{from}\ n^\circ@T\ \mathsf{in}\ P]$ *and M has type $S \not<: T$ where* $\mathsf{filter}_1\ \tilde{x}\ \mathsf{from}\ c@T\ \mathsf{in}\ P$ *is the process reached after unblocking the recursion guard of* $\mathsf{filter}\ \tilde{x}\ \mathsf{from}\ c@T\ \mathsf{in}\ P$ *by means of a deterministic reduction and* $\mathsf{filter}_{2;M}\ \tilde{x}\ \mathsf{from}\ c@T\ \mathsf{in}\ P$ *is the process Q s.t.* $\mathsf{filter}_1\ \tilde{x}\ \mathsf{from}\ c@T\ \mathsf{in}\ P \xrightarrow{c(M)} Q$;

5. $P \equiv C[(\text{new } n^*)n^*\langle N\rangle \,|_{i\in I}\text{if } N_i \notin Set_{n^*} \text{ then } Q_i]$ *and* $P' \equiv C[(\text{new } n^*)(\text{if } N_j \notin N \text{ then } n^*\langle N :: N_j\rangle \,|\, Q_j \text{ else } n^*\langle N\rangle) \,|_{i\in I\setminus j}\text{if } N_i \notin Set_{n^*} \text{ then } Q_i]$ *and* $\forall i,j \in I\,.\,(N_i = N_j) \Rightarrow (Q_i \equiv Q_j)$*;*
6. $P \equiv \text{rec}X.Q$ *and* $P' \equiv C[(\text{new } n)!n(x).Q\{n\langle\rangle/X\} \,|\, Q\{n\langle\rangle/X\}]$*;*
7. $P \equiv C[(\text{new } t)t\langle M\rangle \,|_{i\in I}T_{N_i}(Q_i)]$ *and* $P' \equiv C[(\text{new } t)\text{let } \tilde{y} = ?(N_{jID}, M) \text{ in } t\langle M\rangle \,|\, Q_j \text{ else } ((\text{new } n)Chan_n \,|\, t\langle M.(N_{jID};\underline{n}).(n_{ID};\underline{n})\rangle \,|\, Q_j\{\underline{n}/\tilde{y}\}) \,|_{i\in I\setminus j}T_{N_i}(Q_i)]$
where $T_N(Q) = t(z).\text{let } \tilde{y} = ?(N_{ID}, z) \text{ in } t\langle z\rangle \,|\, Q \text{ else } (\text{new } n)Chan_n \,|\, t\langle z.(N_{ID};\underline{n}).(n_{ID};\underline{n})\rangle \,|\, Q\{\underline{n}/\tilde{y}\}$ *and* $fv(Q) = \tilde{y}$*;*

where $I = 1,\ldots,n$*. We let* $\overset{A}{\Longrightarrow}$ *be the reflexive and transitive closure of* $\overset{A}{\longrightarrow}$*.*

We set $\longrightarrow = \overset{\tau}{\longrightarrow} \setminus \overset{A}{\longrightarrow}$.

Based on this we have the following notion of equivalence that ignore administrative reductions.

Definition 10 (Administrative equivalence). *Administrative equivalence, noted* $\approx^A_\rho$*, is the largest symmetric and contextual term-indexed relation* $\mathscr{R}$ *such that* $\rho \models H\mathscr{R}K$ *implies:*

- *if* $\rho \models H\downarrow_n$ *then* $\rho \models K\downarrow_n$
- $H \overset{A}{\longrightarrow} H'$ *imply* $K \overset{A}{\Longrightarrow} K'$ *for some* K' *such that* $\rho \models H'\mathscr{R}K'$
- $H \longrightarrow H'$ *imply* $K \overset{A}{\Longrightarrow}\longrightarrow\overset{A}{\Longrightarrow} K'$ *for some* K' *such that* $\rho \models H'\mathscr{R}K'$.

A simple, but important property of $\approx^A_\rho$ is that it is contained in $\cong^{A\pi}_\rho$. More precisely:

Lemma 5. *If* $\rho \models H \approx_A K$ *then* $\rho \models H \cong^{A\pi} K$*.*

Proof. Let $\rho \models P\mathscr{R}Q$ whenever $\rho \models P \approx_A Q$. The contextuality of $\mathscr{R}$ follows directly from the definitions (which is the same for both relations). To see that $\mathscr{R}$ preserve barbs, suppose $\rho \models P\downarrow_a$; by definition of $\approx^A_\rho$ we have $\rho \models Q\downarrow_a$ and in turn $\rho \models Q\Downarrow_a$, as requested. To see reduction closure, let $P \overset{\tau}{\longrightarrow} P'$. In case $P \overset{A}{\longrightarrow} P'$ we infer that $\exists Q'$ s.t. $Q \overset{A}{\Longrightarrow} Q'$ and $\rho \models P' \approx_A Q'$. We have thus found Q' s.t. $Q \Longrightarrow Q'$ and $\rho \models P'\mathscr{R}Q'$. In case $P \longrightarrow P'$ we infer that $\exists Q'$ s.t. $Q \overset{A}{\Longrightarrow}\longrightarrow\overset{A}{\Longrightarrow} Q'$ and $\rho \models P' \approx_A Q'$. Thus $Q \overset{\tau}{\Longrightarrow} Q'$ and $\rho \models P'\mathscr{R}Q'$.

We easily obtain that $\approx^A_\rho$ is coarser than $\simeq^{A\pi}_\rho$.

Lemma 6. *If* $\rho \models H \simeq^{A\pi} K$ *then* $\rho \models H \approx^A K$*.*

The key property of administrative reductions is that they are closed under administrative equivalence, in the following sense.

Proposition 7. *Let* ρ *be a term environment s.t.* $fn(H,K) \subseteq fn(\rho)$*. If* $H \overset{A}{\longrightarrow} K$*, then* $\rho \models H \approx^A K$*.*

Proof. (Sketch) We define an asymmetric version of $\approx^A_\rho$, the administrative expansion, noted $\succeq^A_\rho$. Administrative expansion is the largest relation which is strong barb preserving, contextual, and s.t. when $P \succeq^A_\rho Q$ and $P \overset{\tau}{\longrightarrow} P'$ or $Q \longrightarrow Q'$ we have the same clauses of $\approx^A_\rho$, while when $Q \overset{A}{\longrightarrow} Q'$ we have that $P \overset{A^=}{\longrightarrow} P'$ with $Q' \succeq^A_\rho P'$. Then we prove that administrative equivalence up-to administrative expansion, which is defined as the relation $\mathscr{R}$ which have the same clauses of $\approx^A_\rho$ but s.t. the relation reached by reducts is $\succeq^A_\rho\approx^A_\rho\preceq^A_\rho$, is contained in $\approx^A_\rho$.
We Let $\rho \models C[H]\mathscr{R}C[K]$ whenever $\{\!|\,\mathrm{I}\,|\!\} \setminus \tilde{n} \subseteq \rho$ and $C[-] = (\text{new } \tilde{n},\tilde{c})R\rho \,|\, - \wedge\rho \vdash R$ and $H \overset{A}{\longrightarrow} K$. We show that $\mathscr{R} \cup \mathscr{I}$, where $\mathscr{I}$ is the identity relation, is an administrative equivalence up-to administrative expansion.

The following corollary can be proved easily by induction on the number of transitions $\overset{A}{\longrightarrow}$.

Corollary 8. *Let* ρ *be a term environment s.t.* $fn(H,K) \subseteq fn(\rho)$*. If* $H \overset{A}{\Longrightarrow} K$*, then* $\rho \models H \approx^A K$*.*

The following lemma can be easily proved by chasing-diagrams arguments.

Lemma 9. $\simeq^{A\pi}_\rho$, $\approx^A_\rho$ *and* $\cong^{A\pi}_\rho$ *are equivalence relations.*

Protocol Properties Next we need lemmas similar to [3] to ensure that the leftover of the communications of the protocol are invisible to the context; we omit the proofs that are very similar to [3]. We use the following notations. We let $Chan_n\{N\}[Q]$ indicate the process $(\text{new } n^*, n^\circ)\ n^*\langle N\rangle \,|\, Q \,|\, RS_n \,|\, WS_n$ whenever $Q \equiv P_1 \,|\, \cdots \,|\, P_n$ and for all P_i there exists M_i s.t. $WS_n \xrightarrow{net(M_i)} WS \,|\, P_i$ or $RS_n \xrightarrow{net(M_i)} RS_n \,|\, P_i$. We write $Proxy(\tilde{M};\underline{\tilde{s}})\{C\}[S]$ for the process $(\text{new } t)\ P_t \,|\, S \,|\, t\langle\emptyset.(M_{1ID};\underline{s_1}).(s_{1ID};\underline{s_1}).\ldots.(M_{mID};\underline{s_m}).(s_{mID};\underline{s_m})\rangle \,|\, t^*\langle C\rangle$ whenever $\tilde{M} = M_1,\ldots,M_m$, $\tilde{s} = s_1,\ldots,s_m$ and $S \equiv P_1 \,|\, \cdots \,|\, P_n$ and for all P_i there exists M_i s.t. $P_t \xrightarrow{net(M_i)} P_t \,|\, P_i$. We write $M \notin Chan_a\{N\}[P]$ to mean that if $P \equiv a^\circ\langle M_1,\ldots,f(M),\ldots,M_n\rangle \,|\, P'$ then f is a one-way function. We say that $H \xrightarrow{\tau} K$ is an (n,t)-synchronization if the τ step $H \xrightarrow{\tau} K$ derives from $H' \xrightarrow{(\tilde{c})n^\circ\langle\tilde{M}\rangle} K'$, $H'' \xrightarrow{n^\circ(\tilde{M})} K''$, with $H \equiv (\text{new } \tilde{d})H' \,|\, H''$, $K \equiv (\text{new } \tilde{c},\tilde{d})(K' \,|\, K'')$ and M has type t. Typically we note the synchronization steps $\xrightarrow{n@t}$. We let strong behavioural equivalence, noted $\simeq^{A\pi}_{\rho}$, be the largest symmetric and contextual term-indexed relation $\mathscr{R}$ which satisfies the strong version of the clauses of Definition 7.

The first lemma says that a ciphertext cannot be distinguished from noise without knowledge of the symmetric decryption key. Notice that a packet $\{n\}_n$ has no associated destructors, since for symmetric decryption we use the destructor $decipher(cipher(x,sk(y)),sk(y)) \doteq x$.

Lemma 10. *For all* ρ *holds* $\rho \models (\text{new } k)!\,net\langle\{M\}_{sk(k)}\rangle \simeq^{A\pi} (\text{new } n)!\,net\langle\{n\}_n\rangle$.

The next lemma says that the channels do not accept packets with non-fresh nonces and that the ambient see these packets as noise.

Lemma 11. *Let* ρ *be a substitution s.t.* $a, a_w^- \notin A(\rho)$ *and let* $a \neq c$. *For all process* P,Q *s.t.* $\rho \vdash Q$ *and* $c \notin fn(Q)$ *and* $a, a_w^- \notin Chan_a\{N\}[P]$ *we have*

$$\rho \models W \,|\, (\text{new } c)Q\rho \,|\, Chan_a\{N.c\}[P] \,|\, !\,net\langle\{M,c\}_{a_w^+}\rangle \approx^A W \,|\, (\text{new } c)Q\rho \,|\, Chan_a\{N.c\}[P]\ .$$

The next Lemma says that we can remove the nonces from the channel provided they do not appear in the environment.

Lemma 12. *For all processes* P,Q *s.t.* $\rho \vdash Q$ *and* $c \notin fn(P,Q)$ *hold*

$$\rho \models (\text{new } c)Q\rho \,|\, Chan_a\{N.c\}[P] \simeq^{A\pi} (\text{new } c)Q\rho \,|\, Chan_a\{N\}[P]\ .$$

We use a similar Lemma to remove nonces from the Proxy.

Lemma 13. *Let* ρ *be a substitution s.t.* $k, k_P^- \notin A(\rho)$ *and suppose* $\{c,k\} \cap \tilde{s} = \emptyset$. *For all process* S,Q *s.t.* $\rho \vdash Q$ *and* $c \notin fn(Q)$ *we have*

$$\rho \models (\text{new } c)Q\rho \,|\, W \,|\, Proxy(\tilde{M};\underline{\tilde{s}})\{N.c\}[S] \,|\, !net\langle\{\tilde{N},c\}_{k_P^+}\rangle \approx^A (\text{new } c)Q\rho \,|\, W \,|\, Proxy(\tilde{M};\underline{\tilde{s}})\{N.c\}[S]$$

The next lemma says that secret channels are not visible up-to strong barbed congruence.

Lemma 14. *For all* ρ *holds if* $s \notin fn(S)$ *then* $\rho \models (\text{new } s)Chan_s \,|\, Proxy(\tilde{B},B;\tilde{s},s)\{C\}[S] \simeq^{A\pi} Proxy(\tilde{B};\tilde{s})\{C\}[S]$.

Operational Correspondence In this section we establish that the operational correspondence of the centralized encoding $\langle\!\langle\cdot\rangle\!\rangle$. The operational correspondence of $[\![\cdot]\!]$ (Theorem 1) is easier and can be obtained with minor rearrangements. To ease the notation, we let $\{\!|\emptyset|\!\} = net/x, k_P^+/y$, i.e. , for each I we have that Range$(\{\!|\mathrm{I}|\!\})$ contains the proxy key k_P^+. We say that an environment I is compatible with a pi calculus process P if there exists $\Delta <: \mathrm{I}$ s.t. $\Delta \vdash P$.

We first need a Lemma saying that $\langle\!\langle\cdot\rangle\!\rangle$ is closed under substitution, in the following sense.

Lemma 15. $\langle\!\langle P\{v/x\}\rangle\!\rangle \equiv \langle\!\langle P\rangle\!\rangle\{[\![v@A]\!]/\underline{x}\}$.

Lemma 16 (Preservation of execution steps). *Let* I *be compatible with* P. *If* $P \xrightarrow{\tau} P'$ *then* $\mathsf{CE}[\langle\!\langle P\rangle\!\rangle] \xRightarrow{A} \longrightarrow \approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\langle\!\langle P'\rangle\!\rangle]$.

Proof. (Sketch). By induction on the derivation of $P \xrightarrow{\tau} P'$. We sketch the base case (PI-CLOSE@). We prove that $\exists K$ s.t. $\mathsf{CE}[\langle\!\langle P\rangle\!\rangle] \xRightarrow{A} \xrightarrow{n@t} K \approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\langle\!\langle P'\rangle\!\rangle]$ which actually proves the claim as $\xrightarrow{n@t}$ is not administrative. We prove that $K \approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\langle\!\langle P'\rangle\!\rangle]$ by using the lemmas introduced to remove the left over of the communication $\xrightarrow{n@t}$. More in detail, we first

use Lemma 15 to move inside the encoding the capabilities exchanged with $\xrightarrow{n@t}$. Next we use Lemma 10 and $\simeq^{A\pi}_{\{\!|\mathrm{I}|\!\}} \subseteq \approx^{A}_{\{\!|\mathrm{I}|\!\}}$ (Lemma 6 to remove the leftover of the communications protected under a session key. Lemma 11 permit us to remove communications protected under encryption keys possibly known to the context; here we exploit that the protocol always insert a fresh nonce in such packets to prevent cryptanalysis attacks. We apply Lemma 12 to remove nonces from the channel used for matching the pi calculus reduction, i.e. the channel process $Chan_n$. Finally we use Lemma 14 to remove the channel and its entry in the table.

The reverse direction of operational correspondence is subtler, as the encoding is not *prompt*. We therefore need a generalization of the standard reflection result, based on the relation $\approx_A$. First we introduce useful notation and terminology and then state few preliminary lemmas. Let $H \stackrel{A}{\Longrightarrow} H' \longrightarrow K$. We call the reduction sequence $\stackrel{A}{\Longrightarrow}$ *canonical*, and write is as in $\stackrel{\ulcorner A \urcorner}{\Longrightarrow}$, if it only includes the administrative steps from H required to enable the synchronization in H' (as stated, this is loose, but can be made precise, as we know exactly which are those steps).

Lemma 17. *If* $\mathsf{CE}[\langle\!|P|\!\rangle] \stackrel{A}{\Longrightarrow} H \longrightarrow K$ *then there exists* H' *such that* $\mathsf{CE}[\langle\!|P|\!\rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \longrightarrow \bullet \stackrel{A}{\Longrightarrow} K$.

Proof. (Sketch) We first show by case analysis that $H \xrightarrow{n@t} K$. To see that $H \longrightarrow K$ implies $H \xrightarrow{n@t} K$, consider that: (i) H's outputs on net are under replication, while its inputs on net may be under replication, or recursive filters: in the last case they are waiting for terms M encrypted under a session key $sk(N)$ and no other terms $M' \neq M$ are encrypted under $sk(N)$; (ii) reductions on n° or satisfies the expected type or not; (iii) reductions on n^*, t^* are all administrative since translated processes never associate the same nonce to different packets; (iv) reductions on the table channel t satisfy the last clause of Def. 9 for the process $Q \triangleq \mathsf{let}\ \tilde{z} = Cast(\underline{n},t)\ \mathsf{in}\ !\,\{\tilde{z}\}_k$.

Since the names of H are *net*, $\tilde{n}^\circ, \tilde{n}^*, t, t^*$ and "recursion" names $\tilde{r}$ used by processes $\mathsf{rec}X.P$ (which satisfies condition Def. 9(6)), we infer that $H \longrightarrow K$ must be inferred from $H \xrightarrow{n@t} K$. Next we show that a (n,t)-reduction must be preceded by the completion of the write protocol and of the completion of the first part of the read protocol (the asynchronous part). Finally we prove that administrative reductions which are not subsequent (this is the case whenever the first admin. reduction unblocks the next admin. red.) can be "rearranged".

□

Lemma 18. *Let* I *be compatible with* P *and suppose* $\mathsf{CE}[\langle\!|P|\!\rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H \longrightarrow K$. *Then there exists* P' *such that* $P \xrightarrow{\tau} P'$ *and* $\{\!|\mathrm{I}|\!\} \models K \approx_A \mathsf{CE}[\langle\!|P'|\!\rangle]$.

Proof. (Sketch) The lemma follows by exhibiting the canonical administrative post-sequence from K and then by using Lemmas 10,11,12,14 to remove the leftover of the protocol session. □

Lemma 19 (Reflection of execution steps). *Assume* $\{\!|\mathrm{I}|\!\} \models H \approx_A \mathsf{CE}[\langle\!|P|\!\rangle]$ *and* $H \xrightarrow{\tau} K$. *Then either* $\{\!|\mathrm{I}|\!\} \models H \approx_A K$ *or there exists* P' *such that* $P \xrightarrow{\tau} P'$ *and* $\{\!|\mathrm{I}|\!\} \models H \approx_A \mathsf{CE}[\langle\!|P'|\!\rangle]$.

Proof. Let $H \xrightarrow{\tau} K$. If $H \xrightarrow{A} K$, we apply Proposition 7 and we obtain $\{\!|\mathrm{I}|\!\} \models H \approx_A K$. Otherwise $H \longrightarrow K$ and by $\{\!|\mathrm{I}|\!\} \models H \approx_A \mathsf{CE}[\langle\!|P|\!\rangle]$ we infer that there exists Z_0 s.t. $\{\!|\mathrm{I}|\!\} \models \mathsf{CE}[\langle\!|P|\!\rangle] \stackrel{A}{\Longrightarrow}\longrightarrow Z_0 \stackrel{A}{\Longrightarrow} Z$ and $\{\!|\mathrm{I}|\!\} \models K \approx_A Z$. We apply Lemma 17 and we infer that there is Q s.t. $\mathsf{CE}[\langle\!|P|\!\rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow}\longrightarrow Q \stackrel{A}{\Longrightarrow} Z_0$. We apply Lemma 18 ad we obtain that there is P' s.t. $P \xrightarrow{\tau} P'$ and $\{\!|\mathrm{I}|\!\} \models Q \approx^A \mathsf{CE}[\langle\!|P'|\!\rangle]$. From $Q \stackrel{A}{\Longrightarrow} Z_0 \stackrel{A}{\Longrightarrow} Z$ and Corollary 8 we deduce $\{\!|\mathrm{I}|\!\} \models Q \approx^A Z$, and by transitivity of $\approx^A$ (Lemma 9) we obtain $\{\!|\mathrm{I}|\!\} \models Z \approx^A \mathsf{CE}[\langle\!|P'|\!\rangle]$. We use this result and the hypothesis $\{\!|\mathrm{I}|\!\} \models K \approx_A Z$ and transitivity of $\approx^A$ to infer $\{\!|\mathrm{I}|\!\} \models K \approx_A \mathsf{CE}[\langle\!|P'|\!\rangle]$, and we are done.

The following corollary of Lemma 19 will be used to prove the soundness of the translation.

Corollary 20. *If* $\mathsf{CE}[\langle\!|P|\!\rangle] \Longrightarrow H$ *then there exists* P' *such that* $P \Longrightarrow P'$ *and* $\{\!|\mathrm{I}|\!\} \models H \approx_A \mathsf{CE}[\langle\!|P'|\!\rangle]$.

Given a type environment I, to prove preservation and reflection of barbs we define testing contexts of the form $T^n = \mathsf{link}\ (\underline{n},\underline{y})\ \mathsf{in}\ \ (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), \top\}_{y_r^+}) \,|\, \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \omega\langle\rangle$ where $\omega \notin fn(\{\!|\mathrm{I}|\!\})$. The following barb predicate $\mathrm{I} \models P \Downarrow_{n_w^+} \triangleq (n_r^+ \in A(\{\!|\mathrm{I}|\!\}) \wedge \{\!|\mathrm{I}|\!\}, \omega/x \models (T^n \,|\, P) \Downarrow_\omega$ correspond to pi calculus barb $\mathrm{I} \models P \downarrow_n$, in the following sense.

Proposition 21. *The following hold.*

1. $\mathrm{I} \models P\downarrow_a$ *implies* $\mathrm{I} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle]\Downarrow_{a_w^+}$
2. $\mathrm{I} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle]\Downarrow_{a_w^+}$ *implies* $\mathrm{I} \models P\Downarrow_a$.

By using the operational correspondence we prove the only if direction of Theorem 3 and establish the soundness of the translation.

Theorem 22 (Soundness). $\{\!|\mathrm{I}|\!\} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$ *implies* $\mathrm{I} \models P \cong^{\pi} Q$.

Proof Sketch. We consider the relation defined as $\mathrm{I} \vDash P \mathscr{R} Q$ whenever $\{\!|\mathrm{I}|\!\} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$, and we prove that $\mathscr{R}$ is a typed behavioural equivalence. We prove that $\mathscr{R}$ is reduction closed by using Lemma 16 and Corollary 20 (cf. [7]). Let $P \xrightarrow{\tau} P'$; we apply Lemma 16 and find K s.t. $\mathsf{CE}[\langle\!\langle \langle\!\langle P \rangle\!\rangle \rangle\!\rangle] \overset{A}{\Longrightarrow}\longrightarrow K \overset{A}{\Longrightarrow} K_0$ and $\{\!|\mathrm{I}|\!\} \models K \approx^A \mathsf{CE}[\langle\!\langle P' \rangle\!\rangle]$ By Lemma 5 we have $\{\!|\mathrm{I}|\!\} \models K \cong^{A\pi} \mathsf{CE}[\langle\!\langle P' \rangle\!\rangle]$. By reduction closure of $\cong^{A\pi}_{\{\mathrm{I}\}}$ we easily deduce that there exists H s.t. $\mathsf{CE}[\langle\!\langle Q \rangle\!\rangle] \overset{A}{\Longrightarrow}\longrightarrow H$ and $\{\!|\mathrm{I}|\!\} \models K \cong^{A\pi} H$. We apply Corollary 20 to the last weak transition and we obtain that there exists Q' s.t. $Q \Longrightarrow Q'$ and $\{\!|\mathrm{I}|\!\} \models H \approx^A \mathsf{CE}[\langle\!\langle Q' \rangle\!\rangle]$. By Lemma 5 we have $\{\!|\mathrm{I}|\!\} \models H \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q' \rangle\!\rangle]$; by the results above and transitivity of $\cong^{A\pi}_{\{\mathrm{I}\}}$ (Lemma 9) we conclude that $\{\!|\mathrm{I}|\!\} \models \mathsf{CE}[\langle\!\langle P' \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q' \rangle\!\rangle]$, which in turn implies $\mathrm{I} \models P' \mathscr{R} Q'$, as needed.

Barb preservation is obtained directly by using Proposition 21. Let $\mathrm{I} \models P\downarrow_a$; thus $\mathrm{I}(a) <: \mathsf{r}$ and in turn $a_r^+ \in \mathrm{Range}(\{\!|\mathrm{I}|\!\})$. By the contextuality of $\cong^{A\pi}_{\{\mathrm{I}\}}$, we find $\omega \notin fn(\{\!|\mathrm{I}|\!\})$ s.t. $\{\!|\mathrm{I}|\!\}, \omega/x \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$. Now we easily find T s.t. $\{\!|\mathrm{I}|\!\}, \omega/x \vdash T$ and $T^a \equiv T(\{\!|\mathrm{I}|\!\}, \omega/x)$. By closure under parallel compositon of $\cong^{A\pi}$ we infer $\{\!|\mathrm{I}|\!\}, \omega/x \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \,|\, T^a \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle] \,|\, T^a$. By Proposition 21(1) we have $\mathrm{I} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle]\Downarrow_{a_w^+}$; by barb preservation of $\cong^{A\pi}_{\{\mathrm{I}\}}$ it easily follows $\mathrm{I} \models \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]\Downarrow_{a_w^+}$. We apply Prop. 21(2) and we obtain $\mathrm{I} \models Q\Downarrow_a$, as needed.

Finally we show that $\mathscr{R}$ is contextual. Closure under new follows straightforwardly from the closure under new of $\cong^{A\pi}_{\{\mathrm{I}\}}$ and by $\langle\!\langle (\mathsf{new}\ n)P \rangle\!\rangle = (\mathsf{new}\ n)\langle\!\langle P \rangle\!\rangle$. Closure under processes R s.t. $\mathrm{I} \vdash R$ follows by noting that $\langle\!\langle R \rangle\!\rangle$ can be built around $\{\!|\mathrm{I}|\!\}$. To see that $\mathrm{I} \models P \mathscr{R} Q$ implies $\mathrm{I}, a : A \models P \mathscr{R} Q$, we use the contextuality of $\cong^{A\pi}_{\{\mathrm{I}\}}$ to infer $\{\!|\mathrm{I}|\!\}, a/x \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$. Then to add entries $[\![a : A]\!]/x$ to the environment, we use a "weakening" lemma stating that $\rho \vdash M$ and $\rho \models P \cong^{A\pi} Q$ implies $\rho, M/x \models P \cong^{A\pi} Q$. These results let us deduce $\{\!|\mathrm{I}, a : A|\!\} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle]$ and in turn $\mathrm{I}, a : A \models P \cong^{\pi} Q$, as needed. □

Completeness We introduce a encoding which extend $\langle\cdot\rangle$ by allowing encryption keys to be formed around terms. To motivate, notice that both the cryptosystem and the certificates can be instantiated with terms; while client processes build keys and certificates around names, the environment can legitimately create valid encryption keys and certificates around terms.

We extend the encoding of values to arbitrary terms.

$$[\![N@\mathsf{rw}]\!] \triangleq (N_{ID}, N_w^+, N_r^+, N_{CERT}) \qquad [\![N@\mathsf{r}]\!] \triangleq (N_{ID}, hash(N_w^+), N_r^+, N_{CERT})$$
$$[\![N@\mathsf{w}]\!] \triangleq (\mathsf{h}(N), ek(wr(N)), \top)(N_{ID}, N_w^+, hash(N_r^+), N_{CERT}) \qquad [\![N@\top]\!] = (N_{ID}, hash(N_w^+), hash(N_r^+), N_{CERT})$$

We let $\langle\!\langle P \rangle\!\rangle_\beta$, where $\beta = \{(n,N), \dots\}$ is a partial injective function from names to terms (we reserve the name β to this function), be defined as: $\langle\!\langle u\langle v@T \rangle\rangle\!\rangle_{\beta,(v,N)} = (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), \underline{u},\}_{k_p^+}) \,|\, \mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \mathsf{emit}(\{[\![N@T]\!]\}_{y_w^+})$ and $\langle\!\langle u\langle v@T \rangle\rangle\!\rangle_\beta = \langle\!\langle u\langle v@T \rangle\rangle\!\rangle$ whenever $v \notin \mathrm{dom}(\beta)$. We let $\langle\!\langle (\mathsf{new}\ n)P \rangle\!\rangle\beta = (\mathsf{new}\ n)\langle\!\langle P \rangle\!\rangle_\beta$ with $n \notin \mathrm{dom}(\beta)$. The remaining clauses are the same of $\langle\!\langle \cdot \rangle\!\rangle$. We write $\rho \vdash \{\beta, (n,N)\}$ whenever exists N' s.t. $\rho \vdash N' \wedge N = N'\rho$, and $\rho \vdash \beta$. We define $\{\!|\mathrm{I}, a : T|\!\}_{\beta,(a,N)} = \{\!|\mathrm{I}|\!\}_\beta, [\![N@T]\!]/\underline{x}$ and $\{\!|\mathrm{I}, a : T|\!\}_\beta = \{\!|\mathrm{I}|\!\}_\beta, [\![a : T]\!]/\underline{x}$ whenever $a \notin \mathrm{dom}(\beta)$.

The encoding $\langle\!\langle \cdot \rangle\!\rangle_\beta$ is closed under substitution.

Lemma 23. *Let* $\Gamma, x : A \vdash P$ *and* $\Gamma \vdash v : A$. *The following hold.*

1. $\langle\!\langle P\{v/x\} \rangle\!\rangle_{\beta,(v,N)} \equiv \langle\!\langle P \rangle\!\rangle_{\beta,(v,N)}\{[\![N@A]\!]/\underline{x}\}$
2. *if* $v \notin \mathrm{dom}(\beta)$ *then* $\langle\!\langle P\{v/x\} \rangle\!\rangle_\beta \equiv \langle\!\langle P \rangle\!\rangle_\beta\{[\![v@A]\!]/\underline{x}\}$

Preservation of execution steps extend straightforwardly to $\langle\!\langle \cdot \rangle\!\rangle_\beta$.

Lemma 24 (Preservation of execution steps). *Let* I *be compatible with* P, *and suppose* $\{\!|\mathrm{I}|\!\} \vdash \beta$. *If* $P \xrightarrow{\tau} P_1$ *then* $\mathsf{CE}[\langle\!\langle P \rangle\!\rangle_\beta] \overset{A}{\Longrightarrow}\longrightarrow\approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\langle\!\langle P_1 \rangle\!\rangle_\beta]$.

We present now the Theorem that prove that the translation $\langle\!\langle \cdot \rangle\!\rangle_\beta$ is complete. We introduce an up-to technique for behavioural equivalence useful to prove the Theorem.

Lemma 25 (Weak equivalence up-to administrative equivalence). *Let $\mathscr{R}$ be a symmetric indexed relation, contextual and such that $\rho \models H \mathscr{R} K$ implies:*

- *if $\rho \models H \downarrow_n$ then $\rho \models K \Downarrow_n$*
- *$H \stackrel{A}{\Longrightarrow} H'$ implies $K \stackrel{A}{\Longrightarrow} K'$ for some K' such that $\rho \models H' \approx^A \mathscr{R} \approx^A K'$*
- *$H \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} H'$ implies $K \Longrightarrow K'$ for some K' such that $\rho \models H' \approx^A \mathscr{R} \approx^A K'$.*

Then $\mathscr{R} \subseteq \cong^{A\pi}_\rho$.

Theorem 26 (Full Abstraction). *Assume* $\mathrm{dom}(\beta) \subseteq \mathrm{dom}(\mathrm{I})$ *and* $\{\!|\mathrm{I}|\!\} \vdash \beta$. $\mathrm{I} \vDash P \cong Q$ *if and only if* $\{\!|\mathrm{I}|\!\} \models \mathsf{CE}[\langle\!\langle P \rangle\!\rangle_\beta] \cong^{A\pi} \mathsf{CE}[\langle\!\langle Q \rangle\!\rangle_\beta]$.

Proof Sketch. Theorem 22 directly provides the if direction. We therefore prove the only if case.

To ease the notation used in the definition of the candidate relation we consider computing environments of the form $\mathsf{CE}^*{}_\Delta[-] \triangleq W \,|\, Proxy^*_{\Delta_s} \,|\, Chan^*_{\Delta_s}$ representing *configurations* of the proxy, where $\Delta \subseteq \mathrm{I}$. The proxy configuration $Proxy^*_{\Delta_s}$ associates client names in Δ to their server counter-part Δ_s modulo β and contains proxy requests of the context. Formally the proxy map contains entries of the form $(N_{ID}, \underline{\nu_s})$ where or $N = \nu \in \Delta \wedge \nu \notin \mathrm{dom}(\beta)$ or $\beta(\nu) = N$. The scope of Δ_s is outside this definition since the context may have received these names due to a linking request. The channel servers $Chan^*_{\Delta_s}$ created by the proxy contain in their queue both read/write requests submitted by the context and intermediate steps . The computing environment is surrounded by a context representing the low-level knowledge of I; as introduced the context may even contain proxy answers containing capabilities Δ_s. We abuse the notation and write $[\![\Delta_s]\!]$ to indicate the server counter-part of the capabilities $[\![\Delta]\!]_\beta$. To ensure closure under weakening, we augment $\{\!|\mathrm{I}|\!\}$ with fresh names $\tilde{b}$ and we let the index of the relation to be a subset of this base.

Given these intuitions, we let $\rho \models C[\mathsf{CE}^*{}_\Delta[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*{}_\Delta[\langle\!\langle Q \rangle\!\rangle_\beta]])$ whenever

1. $\mathrm{I} \models P \cong Q$ and $\Delta \subseteq \mathrm{I}$
2. $\rho \subseteq (\{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x}) \wedge \tilde{b} \cap fn(\{\!|\mathrm{I}|\!\}) = \emptyset$ and $\mathrm{dom}(\beta) \subseteq \mathrm{dom}(\mathrm{I}) \wedge \{\!|\mathrm{I}|\!\} \vdash \beta$
3. $C[-] = (\mathsf{new}\ \Delta_s, \tilde{c})(R(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \,|\, -)$ and $(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \vdash R$.

We prove that $\mathscr{R}$ is a weak equivalence up-to administrative equivalence. We use the following naming conventions: $C[-]$ is the context, $\mathsf{CE}^*{}_\Delta[-]$ is the environment, $\langle\!\langle P \rangle\!\rangle_\beta$ is the process. For the sake of readability we omit the environment Δ from the definition of $\mathsf{CE}^*{}_\Delta[-]$ whenever no ambiguity may arise. Let $H \stackrel{A}{\Longrightarrow} H' \longrightarrow K$. We call the reduction sequence $\stackrel{A}{\Longrightarrow}$ *canonical*, and write is as in $\stackrel{\ulcorner A \urcorner}{\Longrightarrow}$, if it only includes the administrative steps from H required to enable the synchronization in H' (as stated, this is loose, but can be made precise, as we know exactly which are those steps).

Barb preservation. Let $\rho \models C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \downarrow_a$. If this barb is inferred from $\rho \models \mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta] \downarrow_a$, a case analysis shows that $a = net$. We match this barb with $\rho \models C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Downarrow_a$ inferred from $\rho \models W \downarrow_{net}$. Otherwise $\rho \models C \downarrow_a$ and we trivially have $\rho \models C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Downarrow_a$.

Reduction Closure We need to check the cases $\stackrel{A}{\Longrightarrow}$ and $\stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow}$ of the definition of equivalence up-to administrative equivalence.

Let $C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} K$. By Proposition 7 we have $\{\!|\mathrm{I}|\!\} \models C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \approx^A K$ and we have done since $\rho \models K \approx^A C[\mathsf{CE}^*{}_\Delta[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*{}_\Delta[\langle\!\langle Q \rangle\!\rangle_\beta]])$.

Otherwise let $C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} H \longrightarrow H' \stackrel{A}{\Longrightarrow} K$. We have to found H'' s.t. $C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Longrightarrow H''$ and $\rho \models H' \approx^A \mathscr{R} \approx^A H''$. By closure of $\approx^A$ under administrative reductions and transitivity of $\approx^A_\rho$ this implies $\rho \models K \approx^A \mathscr{R} \approx^A H''$, as requested. The following cases of interaction arise for $H \longrightarrow H'$.

(Context - Environment). We prove that there is a minimal sequence $\stackrel{\ulcorner A\urcorner}{\Longrightarrow}$ and a process H_0 s.t. $C[\mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]] \stackrel{\ulcorner A\urcorner}{\Longrightarrow} C_1[\mathsf{CE}_1^*[\langle\!\langle P\rangle\!\rangle_\beta]] \longrightarrow H_0 \stackrel{A}{\Longrightarrow} H'$ and $H_0 \equiv C_2[\mathsf{CE}_2^*[\langle\!\langle P\rangle\!\rangle_\beta]]$, for some configuration $\mathsf{CE}_2^*[-]$ and context $C_2[-]$ satisfying the definition of $\mathscr{R}$. The intuition is that no prior interaction with the process is needed if $H \longrightarrow H'$ occurred by a communication among the context and the environment. We match this move with $C[\mathsf{CE}^*[\langle\!\langle Q\rangle\!\rangle_\beta]] \stackrel{\ulcorner A\urcorner}{\Longrightarrow} C_1[\mathsf{CE}_1^*[\langle\!\langle Q\rangle\!\rangle_\beta]] \longrightarrow C_2[\mathsf{CE}_2^*[\langle\!\langle Q\rangle\!\rangle_\beta]]$. Next we need to ensure that $C_2[-]$ is a context valid for $\mathscr{R}$. If the synchronization is obtained by the context sending a packet to the environment this clearly holds by the initial hypothesis on $C[-]$. Otherwise the context receives a packet from the environment; a case analysis shows that such packet is a proxy answer containing linked capabilities. Since translated processes do not move, it follows that such request was sended by the context, i.e. there is N and T such that the context knows $[\![N@T]\!]$. From the hypothesis on the shape of $C[-]$ and from $C[-] \stackrel{\ulcorner A\urcorner}{\Longrightarrow} C_1[-]$ and from the communication hypothesis we infer that there is $R_1 \equiv net(x).R' \mid R^*$ closed by $\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}$ such that

$$\begin{array}{rcl} C_1[-] & \equiv & (\mathsf{new}\ \Delta_s, \tilde{c})R_1\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y} \mid - \\ C_2[-] & \equiv & (\mathsf{new}\ \Delta_s, N_s, \tilde{c})(R'\{\{[\![N_s@T]\!]\}_M/x\} \mid R^*)(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \mid - \\ & & \qquad\qquad x \notin \mathrm{dom}(\{\!|\mathrm{I}|\!\}) \cup \{\tilde{x}, \tilde{y}\} \end{array}$$

where N_s is the name indicating the server counterpart of the term N. Three cases arise: (i) $N = v$ for some $v \in \mathrm{dom}(\mathrm{I})$ and $v \notin \mathrm{dom}(\beta)$ or (ii)$\exists v\ .\ \beta(v) = N$ and (iii) $N \in \mathrm{dom}(\mathrm{I})$. In case (i) if $v \in \Delta$ we have done; otherwise we consider $\Gamma = \Delta, v : T$ and we obtain $C_2[-] \equiv (\mathsf{new}\ \Gamma_s, \tilde{c})(R' \mid R^*)(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{x}, [\![N_s@T]\!]/x) \mid -$, as requested. In case (ii) $N_s = v_s$ and we proceed as above and check whether $v \in \Delta$ or not; remember indeed that in this case $[\![v_s@T]\!]$ is the server counterpart of $[\![N@T]\!]$. In case (iii) we choose a name v fresh to I and $\tilde{b}$ and associate it to N in $\beta, (v, N)$. Indeed by weakening closure of $\cong^\pi$ we have $\mathrm{I}, v : \mathsf{rw} \models P \cong^\pi Q$. We obtain $C_2[-] \equiv (\mathsf{new}\ \Gamma_s, \tilde{c})(R' \mid R^*)(\{\!|\mathrm{I}, v : \mathsf{rw}|\!\}, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{x}, [\![v_s@T]\!]/x) \mid -$ where $\Gamma = \Delta, v : \mathsf{rw} \subseteq \mathrm{I}, v : \mathsf{rw}$. Finally from $\rho \subseteq \{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x}$ we obtain $\rho \subseteq \{\!|\mathrm{I}, v : \mathsf{rw}|\!\}, \tilde{b}/\tilde{x}$.

We use the results above and closure of $\approx^A$ under admin. reductions to conclude $\rho \models H' \approx^A C_2[\mathsf{CE}_2^*[\langle\!\langle P\rangle\!\rangle_\beta]] \mathscr{R} C_2[\mathsf{CE}_2^*[\langle\!\langle Q\rangle\!\rangle_\beta]]$.

(Context - Process). By syntactic analysis of the encoding $\langle\!\langle\cdot\rangle\!\rangle_\beta$, we infer that this interaction has occurred on the channel *net*, which is the only free channel used by $\langle\!\langle\cdot\rangle\!\rangle_\beta$. We first analyze the case whether the context inputs a term from the process. We rearrange the administrative reductions $C[\mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} H$ by choosing the minimal sequence which let first the context move and then the process move; the remaining moves are matched with the after-sequence $H_0' \stackrel{A}{\Longrightarrow} H'$:

$$C[\mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]] \stackrel{\ulcorner A\urcorner}{\Longrightarrow} C_1[net(x).R \mid \mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta] \stackrel{\ulcorner A\urcorner}{\Longrightarrow} H_0 \longrightarrow H_0' \stackrel{A}{\Longrightarrow} H'$$

where $H_0 \equiv C_1[net(x).R \mid (\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[P^* \mid net\langle M\rangle]]$ and $H_0' \equiv C_1[(\mathsf{new}\ \tilde{b})R\{M/x\} \mid \mathsf{CE}_1^*[P^*]]$. Here the configuration is possibly evolved to $\mathsf{CE}_1^*[-]$ as the packet received by the context may be a linked emission, that is the proxy could have generated a new channel and association due to a linking request of the process.

A case analysis on $\langle\!\langle\cdot\rangle\!\rangle_\beta$ shows that $P^* \mid net\langle M\rangle \equiv P^*$; this holds since all outputs o *net* syntactically occurring in the encoding are under replication.

We exploit the following fact. Two cases arise for M:

1. $M = \{\tilde{N}\}_{sk(k)}$ and $k \in \tilde{b}$
2. $M = \{\tilde{N}, c\}_{ek(N)}$ and $c \in \tilde{b}$ and $\{dk(N), N\} \cap A(\rho) = \emptyset$

In case (1), we use a Lemma saying that M can be treated as noise whenever k does not occur in the free names of the context (see the Example in Section **??**) and we infer $\rho \models H_0' \simeq^{A\pi} C_1[(\mathsf{new}\ n)R\{\{n\}_n/x\} \mid (\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[P^*]]$. Indeed $k \in \tilde{b}$ implies that $k \notin fn(R)$, i.e. this is a afresh name coming out from the translation. From the hypotheses on the reductions above and from $P^* \mid net\langle M\rangle \equiv P^*$ we obtain $C_1[(\mathsf{new}\ n)R\{\{n\}_n/x\} \mid \mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} C_1[(\mathsf{new}\ n)R\{\{n\}_n/x\} \mid (\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[P^*]]$. We use these results and closure of admin. reductions and $\simeq_\rho^{A\pi} \subseteq \approx_\rho^A$ to infer $H' \approx_\rho^A H_0' \approx_\rho^A C_1[(\mathsf{new}\ n)R\{\{n\}_n/x\} \mid \mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]]$. We match this interaction by letting $net(x).R$ receive a term from the noise process W:

$$\begin{array}{rcl} C[\mathsf{CE}^*[\langle\!\langle Q\rangle\!\rangle_\beta]] & \stackrel{\ulcorner A\urcorner}{\Longrightarrow} & C_1[net(x).R \mid \mathsf{CE}^*[\langle\!\langle Q\rangle\!\rangle_\beta]] \\ & \longrightarrow & C_1[(\mathsf{new}\ n)R\{\{n\}_n/x\} \mid \mathsf{CE}^*[\langle\!\langle Q\rangle\!\rangle_\beta]]\ . \end{array}$$

By $\rho \models H' \approx^A H'_0 \approx^A C_1[(\text{new } n)R\{\{n\}_n/x\} \,|\, \mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C_1[(\text{new } n)R\{\{n\}_n/x\} \,|\, \mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]]$ we obtain reduction closure.

In case (2) we use a Lemma which says that if the confounder c is not in the free names of R and both the decryption key $dk(N)$ and the seed N are not known to context (formally: $\{dk(N), N\} \cap A(\rho) = \emptyset$) then $\rho \models H'_0 \approx^A C_1[(\text{new } n)R\{\{n\}_n/x\} \,|\, (\text{new } \tilde{b})\mathsf{CE}_1^*[P^*]]$. Here R sees M as noise because it has no access to the confounder c and cannot destruct M both by knowing or constructing the key $dk(N)$; however it can administratively forward the packet to the legitimate receiver. This intuition motivates the use of $\approx^A$ in place of the stronger $\simeq^{A\pi}$. As in case (1) we match this interaction by letting $net(x).R$ receive a term from the noise process W:

$$\begin{array}{rcl} C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] & \stackrel{\ulcorner A \urcorner}{\Longrightarrow} & C_1[net(x).R \,|\, \mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \\ & \longrightarrow & C_1[(\text{new } n)R\{\{n\}_n/x\} \,|\, \mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \,. \end{array}$$

We obtain reduction closure with the same equations of case (1).

We now analyze the case whether the process receives a term from the context. We have

$$C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} C_1[net\langle M \rangle \,|\, \mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H_0 \longrightarrow H'_0 \stackrel{A}{\Longrightarrow} H'$$

where $H_0 \equiv C_1[net(x).R \,|\, (\text{new } \tilde{b})\mathsf{CE}_1^*[P^* \,|\, \text{filter}_1\ \tilde{x}\ \text{with } N \text{ in } P]]$ and $H'_0 \equiv C_1[(\text{new } \tilde{b})\ \mathsf{CE}_1^*[P^* \,|\, \text{filter}_{2;M}\ \tilde{x}\ \text{with } N \text{ in } P]]$ where these filters are those introduced in Definition 9.
A case analysis shows that all input on *net* of the translation of P are recursive filters under fresh session keys not known to $C[-]$, i.e. $N = sk(k)$ with $k \in \tilde{b}$. We easily obtain $H'_0 \equiv C_1[net\langle M \rangle \,|\, (\text{new } \tilde{b})\mathsf{CE}_1^*[P^* \,|\, \text{filter } \tilde{x}\ \text{with } N \text{ in } P]]$, i.e. H'_0 reach deterministically in one step H_0, and in turn from Def. 9(1) and closure of $\approx^A$ under admin. reductions we obtain $\rho \models H'_0 \approx^A H_0$. We re-apply closure of $\approx^A$and obtain both $\rho \models H_0 \approx^A C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]]$ and $\rho \models H' \approx^A H'_0$. By applying transitivity of $\approx^A_\rho$ to the results above we have $\rho \models H' \approx^A C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]]$. Summing up: $C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} H' \approx^A_\rho C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]]$. We do not need to match these moves since $\rho \models H' \approx^A C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]]$.

(Environment - Process). There are many cases.

Environment reduction. In this case

$$C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} C[\mathsf{CE}_1^*[\langle\!\langle P \rangle\!\rangle_\beta]] \longrightarrow H'_0 \stackrel{A}{\Longrightarrow} H'$$

A case analysis shows that the synchronization has been inferred from $C[\mathsf{CE}_1^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{n@T}{\longrightarrow} H'_0$ and in turn and $H'_0 \equiv C[!\,\{[\![N@T]\!]\}_M \,|\, \mathsf{CE}_2^*[\langle\!\langle P \rangle\!\rangle_\beta]]$ for some $\mathsf{CE}_2^*[-]$ and N. Indeed the environment $\mathsf{CE}_1^*[-]$ clearly changes it's state since it looses two messages in the queue of $channel_n$. Since the translated process does not move the capabilities $[\![N@T]\!]$ come out from the context and in turn $C[!\,\{[\![N@T]\!]\}_M \,|\, -]$ satifies the definition of $\mathscr{R}$. We match this move with $C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} C[\mathsf{CE}_1^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \longrightarrow C[!\,\{[\![N@T]\!]\}_M \,|\, \mathsf{CE}_2^*[\langle\!\langle Q \rangle\!\rangle_\beta]]$ as $\rho \models H' \approx^A C[!\,\{[\![N@T]\!]\}_M \,|\, \mathsf{CE}_2^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R}$
$C[!\,\{[\![N@T]\!]\}_M \,|\, \mathsf{CE}_2^*[\langle\!\langle Q \rangle\!\rangle_\beta]]$.

Process reduction. In this case

$$C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} \stackrel{n@t}{\longrightarrow} H_0 \stackrel{A}{\Longrightarrow} H'$$

and the synchronization has been inferred from a read and write request of P. We use a minor variant of reflection of execution steps that consider arbitrary environments $\mathsf{CE}^*[-]$ and bindings β to infer that there is P' s.t. $P \stackrel{\tau}{\longrightarrow} P'$ and $\rho \models H_0 \approx^A C[\mathsf{CE}^*[\langle\!\langle P' \rangle\!\rangle_\beta]]$. We exploit the hypothesis $\mathrm{I} \models P \cong Q$ to infer that $Q \Longrightarrow Q'$ with $\mathrm{I} \models P' \cong Q'$. Now we use a variant of preservation of execution steps similar to that above to infer that there is H'' such that $C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Longrightarrow H'' \approx^A_\rho C[\mathsf{CE}^*[\langle\!\langle Q' \rangle\!\rangle_\beta]]$. From closure of $\approx^A$ under admin. reductions and transitivity of $\approx^A$ and the results above we obtain $\rho \models H' \approx^A H_0 \approx^A C[\mathsf{CE}^*[\langle\!\langle P' \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*[\langle\!\langle Q' \rangle\!\rangle_\beta]] \approx^A H''$ and we are done.

Context out - Process in (intrusion). This is the case whereas the encoding of P receives a term from the context trough the queue of a channel in the actual configuration. We have $P \equiv (\text{new } \tilde{b})P_2 \,|\, a(x@T).P_1$ and $\mathrm{I} \vdash a : \mathsf{w}$ and $\mathsf{CE}^*[-] \equiv \mathsf{CE}_1^*[- \,|\, a_1^*\langle N \rangle \,|\, a_1^\circ\langle [\![N@S]\!] \rangle]$ with $S <: T$ and a is linked to a_1. We have the following canonical sequence:

$$\begin{array}{rcl} C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] & \stackrel{\ulcorner A \urcorner}{\Longrightarrow} & C[(\text{new } \tilde{b}, \tilde{c})\mathsf{CE}_2^*[\langle\!\langle P_2 \rangle\!\rangle_\beta \,|\, P^* \,|\, \text{filter } \underline{x}\ \text{with } sk(k) \text{ in } \langle\!\langle P_1 \rangle\!\rangle_\beta]] \\ & \longrightarrow & H'_0 \stackrel{A}{\Longrightarrow} H' \end{array}$$

Here $\tilde{c}$ is formed by the seed k of the shared key $sk(k)$ and the nonce c contained in the input request of the encoding of $a(x@T)$; P^* is the leftover of this request. We have $\mathsf{CE}_2^*[-] \equiv \mathsf{CE}^*{}_{1c}[\mathsf{filter}\ \tilde{x}\ \mathsf{with}\ a_1^\circ@T\ \mathsf{in}\ !net\langle\{\tilde{x}\}_{sk(k)}\rangle]$ where the definition if $\mathsf{CE}^*{}_{1c}$ is the same of CE_1^* but containing the nonce c in the nonce list as in $a_1^*\langle c :: N\rangle$. Finally we have $H_0' \equiv C[(\mathsf{new}\ \tilde{c})\mathsf{CE}_2^*[(\mathsf{new}\ \tilde{b})\langle\!\langle P_2\rangle\!\rangle_\beta \,|\, P^* \,|\, !\,\{\![N@T]\!\}_{sk(k)}\ |\ \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\!\langle P_1\rangle\!\rangle_\beta]]$.

By letting the continuation of P to receive the packet $\{\![N@T]\!\}_{sk(k)}$ and by removing the leftover of the communication we obtain

(i) if $(v,N) \in \beta$ or $N = v$ for some $v \in \mathrm{dom}(\mathrm{I})$ then $H_0' \overset{A}{\Longrightarrow}\approx_\rho^A C[\mathsf{CE}_1^*[(\mathsf{new}\ \tilde{b})\langle\!\langle P_2\rangle\!\rangle_\beta \ |\langle\!\langle P_1\{v/x\}\rangle\!\rangle_\beta]] \equiv C[\mathsf{CE}_1^*[\langle\!\langle(\mathsf{new}\ \tilde{b})P_2 \,|\, P_1\{v/x\}\rangle\!\rangle_\beta]]$

(ii) else there is v s.t. $\{\!|\mathrm{I}, v : S|\!\} \vdash \beta,(v,N)$ and $H_0' \overset{A}{\Longrightarrow}\approx_\rho^A C[\mathsf{CE}_1^*[(\mathsf{new}\ \tilde{b})\langle\!\langle P_2\rangle\!\rangle_\beta \ |\langle\!\langle P_1\{v/x\}\rangle\!\rangle_{\beta,(v,N)}]] \equiv C[\mathsf{CE}_1^*[\langle\!\langle(\mathsf{new}\ \tilde{b})P_2 \,|\, P_1\{v/x\}\rangle\!\rangle_{\beta,(v,N)}]]$ with $v \notin n(P)$.

By the hypothesis $\mathrm{I} \models P \cong Q$ we infer that in case (i) $\mathrm{I} \models a\langle v@S\rangle \,|\, P \cong a\langle v@S\rangle \,|\, Q$ and in case (ii) $\mathrm{I}, v : S \models a\langle v@S\rangle \,|\, P \cong a\langle v@S\rangle \,|\, Q$. Since $a\langle v@S\rangle \,|\, P \xrightarrow{\tau} P' \triangleq (\mathsf{new}\ \tilde{b})P_2 \,|\, P_1\{v/x\}$, by hypothesis there is Q' s.t. $a\langle v@S\rangle \,|\, Q \Longrightarrow Q'$ and (i) $\mathrm{I} \models P' \cong Q'$ or (ii) $\mathrm{I}, v : S \models P' \cong Q'$. We apply preservation of execution steps to process $a\langle v@S\rangle \,|\, Q$ immerged in the environment $\mathsf{CE}_1^*[-]$, i.e. the environment not containing the message $a_1\langle N@T\rangle$, and obtain

$$\text{(i)} \quad C[\mathsf{CE}_1^*[\langle\!\langle a\langle v@S\rangle \,|\, Q\rangle\!\rangle_\beta]] \Longrightarrow\approx_\rho^A C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_\beta]]$$

$$\text{(ii)} \quad C[\mathsf{CE}_1^*[\langle\!\langle a\langle v@S\rangle \,|\, Q\rangle\!\rangle_{\beta,(v,N)}]] \Longrightarrow\approx_\rho^A C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_{\beta,(v,N)}]]$$

It's easy to see that in case (i) $C[\mathsf{CE}_1^*[\langle\!\langle a\langle v@S\rangle \,|\, Q'\rangle\!\rangle_\beta]] \overset{A}{\Longrightarrow}\approx_\rho^A C[\mathsf{CE}^*[\langle\!\langle Q'\rangle\!\rangle_\beta]]$ and in case (ii) $C[\mathsf{CE}_1^*[\langle\!\langle a\langle v@S\rangle \,|\, Q'\rangle\!\rangle_{\beta,(v,N)}]] \overset{A}{\Longrightarrow}\approx_\rho^A C[\mathsf{CE}^*[\langle\!\langle Q'\rangle\!\rangle_{\beta,(v,N)}]]$; in these results we use $\approx^A$ to discard unuseful leftoftvers of the write request.
In case (i) from closure of $\approx^A$ under administrative reductions we infer
$C[\mathsf{CE}_1^*[\langle\!\langle a\langle v@S\rangle \,|\, Q'\rangle\!\rangle_\beta]] \approx_\rho^A C[\mathsf{CE}^*[\langle\!\langle Q'\rangle\!\rangle_\beta]]$ and in turn from the results above

$$C[\mathsf{CE}^*[\langle\!\langle Q'\rangle\!\rangle_\beta]] \Longrightarrow H'' \approx_\rho^A C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_\beta]]\ .$$

Similarly in case (ii) we deduce that there is H'' such that

$$C[\mathsf{CE}^*[\langle\!\langle Q'\rangle\!\rangle_{\beta,(v,N)}]] \Longrightarrow H'' \approx_\rho^A C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_{\beta,(v,N)}]]\ .$$

From closure of $\approx^A$ under admin. reductions we infer (i) $\rho \models H' \approx^A C[\mathsf{CE}_1^*[\langle\!\langle P'\rangle\!\rangle_\beta]]$ or (ii) $\rho \models H' \approx^A C[\mathsf{CE}_1^*[\langle\!\langle P'\rangle\!\rangle_{\beta,(v,N)}]]$. We have done as in case (i) we have $\rho \models H' \approx^A C[\mathsf{CE}_1^*[\langle\!\langle P'\rangle\!\rangle_\beta]]\mathscr{R}C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_\beta]] \approx^A H''$ and in case (ii) we have $\rho \models H' \approx^A C[\mathsf{CE}_1^*[\langle\!\langle P'\rangle\!\rangle_{\beta,(v,N)}]]\mathscr{R}C[\mathsf{CE}_1^*[\langle\!\langle Q'\rangle\!\rangle_{\beta,(v,N)}]] \approx^A H''$. In (ii) notice that $\rho = \sigma, \tilde{b}/\tilde{x}$ and $\sigma \subseteq \{\!|\mathrm{I}, v : S|\!\}$ have been deduced from the hypothesis $\sigma \subseteq \{\!|\mathrm{I}|\!\}$.

Context in - Process out (extrusion). In this case the process sends capabilities to the context trough the environment. We have $P \equiv (\mathsf{new}\ \tilde{b} : \tilde{B})P' \,|\, a\langle v@S\rangle$ and $\mathrm{I} \vdash a : \mathsf{r}$ and $\mathsf{CE}[-] \equiv (\mathsf{new}\ \tilde{a})\mathsf{CE}_1^*[a_1^*\langle N\rangle \,|\, \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ a_1^\circ@T\ \mathsf{in}\ !net\langle\{\tilde{x}\}_M\rangle]$ where a is linked to a_1 and $S <: T$. We have the following canonical sequence:

$$C[\mathsf{CE}^*[\langle\!\langle P\rangle\!\rangle_\beta]] \overset{\ulcorner A\urcorner}{\Longrightarrow} C[(\mathsf{new}\ \tilde{b}, \tilde{c})\mathsf{CE}^*{}_{1c}[a_1^\circ\langle[\![N@S]\!]\rangle \,|\, \langle\!\langle P'\rangle\!\rangle_\beta \,|\, P^*]] \longrightarrow H_0' \overset{A}{\Longrightarrow} H'$$

where P^* is the leftover of the communication to send the output request containing the nonce $c = \tilde{c}$ and $\mathsf{CE}^*{}_{1c}[-]$ is defined as $\mathsf{CE}^*{}_1[-]$ but contains c in the nonce list and $(v,N) \in \beta$. The case $N = v \in \mathrm{dom}(\mathrm{I})$ is simpler and we omit it. The synchronization involves the filter receiving the packet $[\![N@S]\!]$ on a_1°. By removing the leftover we have

$$H_0' \approx_\rho^A C[(\mathsf{new}\ \tilde{b})!net\langle\{[\![N@T]\!]\}_M\rangle \,|\, \mathsf{CE}_2^*[\langle\!\langle P'\rangle\!\rangle_\beta]]\ .$$

We exploit the characterization of $\cong^{A\pi}_{\{\mathrm{I}\}}$ in terms of typed bisimulation presented in Chapter **??** and we infer that: $\mathrm{I} \vDash P \cong Q$, $\mathrm{I} \triangleright P \xrightarrow{(\tilde{b})a\langle v@T\rangle} \mathrm{I} \sqcap v : T \triangleright P'$, implies that thre is Q' such that$\mathrm{I} \triangleright Q \xRightarrow{(\tilde{b})a\langle v@T\rangle} \mathrm{I} \sqcap v : T \triangleright Q'$ with $\mathrm{I} \sqcap v : T \models P' \cong Q'$. Thus exists Q_0, s.t. $Q \Longrightarrow Q_0$ and $\mathrm{I} \triangleright Q_0 \xrightarrow{(\tilde{b})a\langle v@T\rangle} \mathrm{I} \sqcap v : T \triangleright Q_0'$ and $Q_0' \Longrightarrow Q'$. From these results we infer that $Q_0 \equiv (\mathsf{new}\ \tilde{b})Q_0' \,|\, a\langle v@T\rangle$.

We apply preservation of execution steps and infer $C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Longrightarrow \approx^A_\rho$
$C[\mathsf{CE}^*[\langle\!\langle Q_0 \rangle\!\rangle_\beta]]$. From the shape of Q_0 and by following the same steps we did in showing that $C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} C[(\mathsf{new}\ \tilde{b}, \tilde{c})$ $\mathsf{CE}^*{}_{1c}[\langle\!\langle P_1 \rangle\!\rangle_\beta \,|\, P^*]]$ we infer

$$C[\mathsf{CE}^*[\langle\!\langle Q_0 \rangle\!\rangle_\beta]] \stackrel{A}{\Longrightarrow} C[(\mathsf{new}\ \tilde{b}, \tilde{c})\mathsf{CE}^*{}_{1c}[\langle\!\langle Q_0' \rangle\!\rangle_\beta \,|\, P^*]] \,.$$

Notice indeed that both the leftover P^* and the environment $\mathsf{CE}^*{}_{1c}[-]$ are determined by the (encoding of) $a\langle v@T \rangle$ which is the same for both processes (up-to alpha-renaming of bindings $\tilde{b}, \tilde{c}$). Now we let the synchronization occur inside the environment $\mathsf{CE}^*{}_{1c}[-]$ and we remove the leftover:

$$\begin{aligned} C[(\mathsf{new}\ \tilde{b}, \tilde{c})\mathsf{CE}^*{}_{1c}[\langle\!\langle Q_0' \rangle\!\rangle_\beta \,|\, P^*]] &\to K_1 \\ &\approx^A_\rho C[(\mathsf{new}\ \tilde{b})!net\langle\{\!|N@T|\!\}_M\rangle \,|\, \mathsf{CE}^*_2[\langle\!\langle Q_0' \rangle\!\rangle_\beta]] \,. \end{aligned}$$

Finally we apply preservation of execution steps and reduction closure of $\approx^A$ and obtain

$$K_1 \Longrightarrow K'' \approx^A_\rho C[(\mathsf{new}\ \tilde{b})!net\langle\{\!|N@T|\!\}_M\rangle \,|\, \mathsf{CE}^*_2[\langle\!\langle Q' \rangle\!\rangle_\beta]] \,.$$

Summing up we have:

$$C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]] \Longrightarrow K'' \approx^A_\rho C[(\mathsf{new}\ \tilde{b})!net\langle\{\!|N@T|\!\}_M\rangle \,|\, \mathsf{CE}^*_2[\langle\!\langle Q' \rangle\!\rangle_\beta \,|\, P^*]]$$

By definition of $\mathscr{R}$ we have that $C[-] \equiv (\mathsf{new}\ \tilde{c})R(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \,|-$ where $(\{\!|\mathrm{I}|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \vdash R$. Notice that the bindings $\tilde{c}$ may contain names in I that do not occur in the base. We define $D[-] = (\mathsf{new}\ \tilde{b})C[R_1(\{\!|\mathrm{I} \sqcap v : T|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \,|-]$ where $(\{\!|\mathrm{I} \sqcap v : T|\!\}_\beta, \tilde{b}/\tilde{x}, [\![\Delta_s]\!]/\tilde{y}) \vdash R_1$. It's easy to find R_1 s.t.
$C[(\mathsf{new}\ \tilde{b})!net\langle\{\!|N@T|\!\}_M\rangle \equiv D[-]$. Indeed we have what we need to build the term $[\![N@T]\!]$, and M was previously used by the environment.
Since $\mathrm{I} \sqcap v : T \models P' \cong Q'$ and $(v, N) \in \beta$ we have

$$\rho \models H' \approx^A H_0' \approx^A D[\mathsf{CE}^*_2[\langle\!\langle P' \rangle\!\rangle_\beta]] \mathscr{R} D[\mathsf{CE}^*_2[\langle\!\langle Q' \rangle\!\rangle_\beta]] \approx^A H''$$

as desired.

Contextuality Let $\rho \models C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]])$. For the first clause, let $\rho \vdash S$. By $\rho \subseteq (\{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x})$ we have $\rho \models C[S\rho \,|\, \mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[S\rho \,|\, \mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]])$ where $fn(S\rho) \cap bn(C) = \emptyset$. For the second clause, let $n \notin fn(\rho)$. Since $\rho \subseteq \{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x}$, we have $\rho, n/x \subseteq \{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x}, n/x$ provided $n \notin \tilde{b}$.. We have $\rho, n/x \models C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]])$ as $C \equiv C\{n/x\}$ holds since C has not free variables. Last, we analyze the third clause. From $\rho \setminus n \subseteq (\{\!|\mathrm{I}|\!\}, \tilde{b}/\tilde{x})$ we infer $\rho \setminus n \models (\mathsf{new}\ n)C[\mathsf{CE}^*[\langle\!\langle P \rangle\!\rangle_\beta]] \mathscr{R} (\mathsf{new}\ n)C[\mathsf{CE}^*[\langle\!\langle Q \rangle\!\rangle_\beta]])$ as $(\mathsf{new}\ n)C[-]$ satisfies the definition required in $\mathscr{R}$ by the hypothesis on $C[-]$.

□