A Static Analysis of the Applied Pi Calculus

Benjamin Aziz

Department of Computing
Imperial College London
180 Queen's Gate, London SW7 2RH, UK

Abstract

We present in this technical report a non-uniform static analysis for detecting the term-substitution property in systems specified in the language of the applied pi calculus. The analysis implements a denotational framework that has previously introduced analyses for the pi calculus and the spi calculus. The main novelty of this analysis is its ability to deal with systems specified in languages with non-free term algebras, like the applied pi calculus, where non-identity equations may relate different terms of the language. We demonstrate the applicability of the analysis to one famous security protocol, which uses non-identity equations, namely the Diffie-Hellman protocol.

1 Introduction

In previous works [9, 10, 11, 12], we defined a non-uniform static analysis framework for capturing the property of term substitutions in processes modelled by the pi calculus [26] and the spi calculus [6]. We demonstrated that this property was essential in understanding certain types of secrecy and authenticity threats, in particular, whenever processes classified at low secrecy levels obtained high-level data, or when highly trusted processes obtain low-level data.

In this report, we extend the static analysis framework to include systems modelled in the applied pi calculus [5]. The applied pi calculus was introduced primarily as a generalisation of the spi calculus incorporating non-free term algebras, where, non-identity equations may exist relating the different terms of the language. Hence, in the spi calculus, the only equations available are of the form $M = M$, and therefore, it is not possible to handle protocols with terms related by equations like $f(x, g(y)) = f(y, g(x))$. Such equations are popular in security protocols and form the basis, for example, of all the Diffie-Hellman protocol variants. As a result, the applied pi calculus equips terms with general equational theories that can either reshape or deconstruct the original term. Moreover, the applied pi calculus emphasizes the *openness* of processes by giving their contexts the ability to perform active substitutions (i.e. substitutions on the fly). This latter feature of the language seems to render the static analysis of [10] a suitable framework based on capturing the term substitutions property.

The area of the static analysis of cryptographic processes has been researched intensively in recent years using a variety of techniques, for which one can only mention a few examples here. These include *types* [1, 2, 13], *symbolic methods* [8, 18, 19, 23, 25], *abstract interpretation* [9, 10, 12, 27], *control flow analysis* [14, 15, 16, 17, 32] and other techniques [3, 4, 13, 22]. In particular, [4] uses a specific variation of the applied pi calculus to model *Just-Fast-Keying*, a protocol for the establishment of secure session keys using Diffie-Hellman exchanges and public-key cryptography. Also, [22] gives an analysis for the authentication properties of a protocol similar to SKEME [24], however, the analysis is based on equivalence relations and is not automatic. Finally, [32] constructs a control-flow analysis of a primitive version of the applied pi calculus with only free term algebra. The analysis is uniform in its term capturing and cannot therefore distinguish properties that may change with the different runs of a protocol.

The rest of the report is structured as follows. In Section 2, we review the syntax of the applied pi language and in Section 3, we develop a domain-theoretic model, which is used then to define a standard denotational semantics for the applied pi calculus. The standard semantics is extended in Section 4 to capture the property of term substitutions and in Section 5, we define an abstraction, which safely constrains the size of the semantic domain to a finite limit. In Section 6, we define secrecy and authenticity properties and in Section 7, we apply the analysis to the Diffie-Hellman protocol. Finally, in Section 8, we conclude the report and give directions for future work.

2 The Applied Pi Calculus

The syntax of the applied pi calculus is summarized in Figure 1.

$L, M, N, T, U, V \in \mathcal{T}$	$::=$	Terms
	$a, b, c, \ldots, s \in \mathcal{N}$	Names
	$x, y, z \in \mathcal{V}$	Variables
	$f(M_1, \ldots, M_n)$	Function application
$P, Q, R \in \mathcal{P}$	$::=$	Processes
	$\mathbf{0}$	Null process
	$P \mid Q$	Parallel composition
	$!P$	Replication
	$\nu n.P$	Name restriction
	if $M = N$ *then* P *else* Q	Conditional
	$M(x).P$	Input
	$\overline{M}\langle N\rangle.P$	Output
$A, B, C \in \mathcal{EP}$	$::=$	Extended processes
	P	Process
	$A \mid B$	Parallel composition
	$\nu n.A$	Name restriction
	$\nu x.A$	Variable restriction
	$\{M/x\}$	Active substitution

Figure 1: The syntax of the applied pi calculus.

This syntax is similar to that of [5] with which we assume the reader to be familiar. Functions are assumed to be taken from a finite signature, Σ, equipped with an equational theory that is used to infer when two terms are equal, $\Sigma \vdash M = L$, such as for example, $\mathtt{decrypt}(\mathtt{encrypt}(x, y), y) = x$. We also assume that the usual notions of free and bound names and variables of terms, processes and extended processes, as well as α-conversion all apply. Initially, we require that α-conversion be used to achieve the following property on extended processes.

Property 1 *For any extended process, $A \in \mathcal{EP}$, there are no occurrences of homonymous bound names and variables, such as for example, $a(x).P \mid b(x).Q$, or $\nu n.A \mid \nu n.B$. Moreover, the following holds:*
$\forall s, s' \in \{bn(A), fn(A), bv(A), fv(A)\} : s \cap s' = \{\}$.

We also define the set of bound names and variables for any entity, e, as $bnv(e) = bn(e) \cup bv(e)$, and the set of free names and variables of e as $fnv(e) = fn(e) \cup fv(e)$.

The structural operational semantics of the applied pi calculus is defined in terms of the structural congruence ($\equiv$) relation (Figure (2)) and reaction ($\xrightarrow{a\pi}$) relation (Figures (3). In this semantics, we have replaced early input transitions, $A \xrightarrow{a(M)} A'$ as they appeared in [5, §4.4], with the late version, $A \xrightarrow{a(x)} A'$. This implies that input actions are only be instantiated through reductions, $\overline{a}\langle x\rangle.P \mid a(x).Q \xrightarrow{\tau} P \mid Q$, and active substitutions, $\{M/x\}$.

PAR-0	A	$\equiv$	$A \mid 0$
PAR-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$
PAR-C	$A \mid B$	$\equiv$	$B \mid A$
REPL	$!P$	$\equiv$	$P \mid !P$
NEW-0	$\nu n.0$	$\equiv$	0
NEW-C	$\nu u.\nu v.A$	$\equiv$	$\nu v.\nu u.A$
NEW-PAR	$A \mid \nu v.B$	$\equiv$	$\nu u.(A \mid B)$ *when,* $u \notin fv(A) \cup fn(A)$
ALIAS	$\nu x.\{M/x\}$	$\equiv$	0
SUBST	$\{M/x\} \mid A$	$\equiv$	$\{M/x\} \mid A\{M/x\}$
REWRITE	$\{M/x\}$	$\equiv$	$\{N/x\}$ *when,* $\Sigma \vdash M = N$

Figure 2: Rules of the structural congruence relation, $\equiv$.

COMM	$\overline{a}\langle x\rangle.P \mid a(x).Q \xrightarrow{\tau} P \mid Q$		
THEN	*if* $M = M$ *then* P *else* $Q \xrightarrow{\tau} P$		
ELSE	*if* $M = N$ *then* P *else* $Q \xrightarrow{\tau} Q$ *for any ground terms* M *and* N *such that* $\Sigma \nvdash M = N$		
LATE-IN	$a(x).P \xrightarrow{a(x)} P$		
OUT-TERM	$\overline{a}\langle M\rangle.P \xrightarrow{\overline{a}\langle M\rangle} P$		
LIM-SCOPE	$A \xrightarrow{a\pi'} A'$ *where,* $a\pi' \in \{a(x), a\langle M\rangle, \tau\}$ *and,* $u \notin n(a\pi')$	$\Rightarrow$	$\nu u.A \xrightarrow{a\pi'} \nu u.A'$
PAR	$A \xrightarrow{a\pi} A' \wedge bv(a\pi) \cap fv(B) = bn(a\pi) \cap fn(B) = \{\}$	$\Rightarrow$	$A \mid B \xrightarrow{a\pi} A' \mid B$
OPEN-CHANNEL	$A \xrightarrow{\overline{a}\langle b\rangle} A' \wedge b \neq a$	$\Rightarrow$	$\nu b.A \xrightarrow{\nu b.\overline{a}\langle b\rangle} A'$
OPEN-VARIABLE	$A \xrightarrow{\nu\, u_1 \ldots \nu\, u_k.\overline{a}\langle M\rangle} A' \wedge x \in fv(M)\backslash\{u_1, \ldots, u_k\} \wedge$ *x can be derived from* $(\nu u_1 \ldots \nu u_k.\{M/z\} \mid A')$	$\Rightarrow$	$\nu x.A \xrightarrow{\nu x.\nu\, u_1 \ldots \nu\, u_k.\overline{a}\langle M\rangle} A'$
STRUCT	$A \equiv B \wedge B \xrightarrow{a\pi} B' \wedge B' \equiv A'$	$\Rightarrow$	$A \xrightarrow{a\pi} A'$

Figure 3: Rules of the refined late labelled transition relation, $\xrightarrow{a\pi}$.

3 A Domain-Theoretic Model

Following the approach of [9, 10, 11, 12, 31], we define the following predomain equations, which describe what an extended process can do:

$$\begin{aligned}
Api &\cong 1 + \\
&\quad \mathbb{P}(Api_\perp + In + Out + Sub) && (1)\\
In &\cong Pmv \times (T \to Api_\perp) && (2)\\
Out &\cong Pmv \times (T \times Api_\perp + Pmv \to \ldots \\
&\quad Pmv \to (T \times Api_\perp)) && (3)\\
Sub &\cong Api_\perp \to (Api_\perp + Pmv \to \ldots \\
&\quad Pmv \to Api_\perp) && (4)\\
T &\cong Pmv + Cpx && (5)\\
Pmv &\cong N + V && (6)\\
Cpx &\cong T \times \ldots \times T && (7)
\end{aligned}$$

Where $Api_\perp$ is the domain of extended processes, *In*, *Out* and *Sub* are the predomains of input/output actions and active substitutions, respectively. $\mathbb{P}(-)$ is Plotkin's powerdomain [29] applied to the disjoint union of input/output/silent actions (the latter represented by $Api_\perp$) and active substitutions to construct *Api*. The single element predomain, 1, representing terminated (deadlocked) processes is adjoined as in [7, Def. 3.4]. Input actions are modelled as pairs; a primitive term (the channel), *Pmv*, and a function, $T \to Api_\perp$, that can be

instantiated with a term, T, yielding a process in $Api_\perp$. Output actions are divided into free and bound output actions. These are pairs consisting of a primitive term (the channel), Pmv, and either another pair, $T \times Api_\perp$, denoting the message, T, and the residue, $Api_\perp$ (free outputs), or composed functions, $Pmv \to \ldots Pmv \to (T \times Api_\perp)$, that introduce new primitive terms to the message, T, and the residue, $Api_\perp$ (bound outputs). These functions express the output of a complex term with several bound names or variables. Active substitutions are modelled as a function which accepts an extended process, $Api_\perp$, and returns either another extended process, $Api_\perp$ (free substitution), or a number of composed functions leading to an extended process, $Pmv \to \ldots Pmv \to Api_\perp$ (bound substitution, e.g. $A \stackrel{\mathtt{def}}{=} \nu n_1 \ldots \nu n_k.\{M/x\}$, where $n_1 \ldots n_k \in fn(M)$). The predomain of terms, T, is defined as the disjoint union of the flat predomains of primitive terms, Pmv, and complex terms, Cpx. Primitive terms are either names, N (flat predomain), or variables, V (flat predomain), whereas complex terms are essentially tuples, $T \times \ldots \times T$, that cannot be reduced to elements of N or V under Σ.

The main difference between the above equations and the predomain equations of [10] is the presence of the predomain of active substitutions, Sub, and the generalisation of complex terms as tuples. Also, the predomain of variables, V, permits dealing with open terms and open extended processes.

The following functions are defined as usual [7, Def. 3.3], leading to $Api_\perp$:

$$\begin{aligned}
\emptyset &: 1 \to Api_\perp && (8)\\
\{\!|-|\!\} &: (Api_\perp + In + Out + Sub)_\perp \to \\
&\quad Api_\perp && (9)\\
\uplus &: (Api_\perp \times Api_\perp) \to Api_\perp && (10)\\
new &: (Pmv \to Api_\perp) \to Api_\perp && (11)
\end{aligned}$$

The empty set, $\emptyset$, is required to represent inactive processes. The singleton map, $\{\!|-|\!\}$, creates elements of $Api_\perp$ from elements of input/output/silent actions and active substitutions. $\uplus$, is the standard multiset union operator representing non-determinism. Finally, new is used to interpret the effects of restricting a primitive term to the scope of an extended process.

Next, we define concrete elements of the (pre)domains of equations (1)–(7), as in Figure 4, where $\mathcal{K}$ is the set of elements underlying any (pre)domain. This definition leads ultimately to the definition of elements $p, q \in Api_\perp$. Note that, for simplicity, we assume that elements $t, t' \in T$ are taken directly from the set of syntactic terms, $\mathcal{T}$. Also, we have $\{\!|\perp|\!\} \sqsubseteq \emptyset$.

The effects of restriction are interpreted by concretely defining new as in Figure 5. These effects lead to the blocking of processes attempting to communicate over fresh, non-extruded channels and the transformation of free outputs to bound outputs whenever the message of communication is restricted in some (all) of its primitive terms. Similarly, substitutions involving restricted variables are blocked. Alternatively, if the primitive term being restricted belongs to the free names and variables of the substituting term, then the substitution

$$
\begin{array}{lcl}
\textit{Elements of } N,\ V \textit{ and } Cpx: & & \\
n \in \mathcal{N} & \Rightarrow & n \in \mathcal{K}(N) \\
x \in \mathcal{V} & \Rightarrow & x \in \mathcal{K}(V) \\
f \in \Sigma, t_1, \ldots, t_{ar(f)} \in T, \nexists u \in Pmv: & & \\
\quad \Sigma \vdash\ f(t_1, \ldots, t_{ar(f)}) = u & \Rightarrow & f(t_1, \ldots, t_{ar(f)}) \in \mathcal{K}(Cpx) \\
 & & \\
\textit{Elements of } In: & & \\
u \in \mathcal{K}(Pmv), \lambda y.p \in \mathcal{K}(T \rightarrow Api_\perp) & \Rightarrow & (u, \lambda y.p) \in \mathcal{K}(In) \\
 & & \\
\textit{Elements of } Out: & & \\
u \in \mathcal{K}(Pmv), t \in \mathcal{K}(T), p \in \mathcal{K}(Api_\perp) & \Rightarrow & (u, t, p) \in \mathcal{K}(Out) \\
u \in \mathcal{K}(Pmv), \lambda u_1 \ldots \lambda u_m.(t, p) \in & & \\
\mathcal{K}(Pmv \rightarrow_1 \ldots Pmv \rightarrow_m (T \times Api_\perp)) & \Rightarrow & (u, \lambda u_1 \ldots \lambda u_m.(t, p)) \in \mathcal{K}(Out) \\
 & & \\
\textit{Elements of } Sub: & & \\
x \in \mathcal{K}(V), t \in \mathcal{K}(T), \lambda y.((\lambda x.y)t) \in \mathcal{K}(Api_\perp \rightarrow Api_\perp) & \Rightarrow & \lambda y.((\lambda x.y)t) \in \mathcal{K}(Sub) \\
x \in \mathcal{K}(V), t \in \mathcal{K}(T), \lambda y.((\lambda x.\lambda u_1 \ldots \lambda u_k.y)t) \in & & \\
\mathcal{K}(Api_\perp \rightarrow Pmv \rightarrow_1 \ldots \rightarrow_m Pmv \rightarrow Api_\perp) & \Rightarrow & \lambda y.((\lambda x.\lambda u_1 \ldots \lambda u_k.y)t) \in \mathcal{K}(Sub) \\
 & & \\
\textit{Elements of } Api_\perp: & & \\
\{\!|\perp|\!\} \in \mathcal{K}(Api_\perp), \emptyset \in \mathcal{K}(Api_\perp) & & \\
p, q \in \mathcal{K}(Api_\perp) & \Rightarrow & p \uplus q \in \mathcal{K}(Api_\perp) \\
p \in \mathcal{K}(Api_\perp) & \Rightarrow & \{\!|tau(p)|\!\} \in \mathcal{K}(Api_\perp) \\
i \in \mathcal{K}(In) & \Rightarrow & \{\!|in(i)|\!\} \in \mathcal{K}(Api_\perp) \\
o \in \mathcal{K}(Out) & \Rightarrow & \{\!|out(o)|\!\} \in \mathcal{K}(Api_\perp) \\
s \in \mathcal{K}(Sub) & \Rightarrow & \{\!|s|\!\} \in \mathcal{K}(Api_\perp)
\end{array}
$$

Figure 4: Elements of the domain $Api_\perp$.

$$
\begin{array}{l}
new(\lambda u.\emptyset) \ =\ \emptyset \\
new(\lambda u.\{\!|\perp|\!\}) \ =\ \{\!|\perp|\!\} \\
new(\lambda u.\{\!|in(u', \lambda x.p)|\!\}) \ =\ \begin{cases} \emptyset, & \text{if } u = u' \\ \{\!|in(u', \lambda x.new(\lambda u.p))|\!\}, & \text{otherwise} \end{cases} \\
new(\lambda u.\{\!|out(u', t, p)|\!\}) \ =\ \begin{cases} \emptyset, & \text{if } u = u' \\ \{\!|out(u', \lambda u.(t, p))|\!\}, & \text{if } u \in fnv(t) \ \wedge\ u \neq u' \\ \{\!|out(u', t, new(\lambda u.p))|\!\}, & \text{otherwise} \end{cases} \\
new(\lambda u.\{\!|out(u', \lambda u_1 \ldots \lambda u_k.(t, p))|\!\}) \ = \\
\begin{cases} \emptyset, & \text{if } u = u' \\ \{\!|out(u', \lambda u.\lambda u_1 \ldots \lambda u_k.(t, p))|\!\}, & \text{if } u \in fnv(t) \ \wedge\ u \neq u' \\ \{\!|out(u', \lambda u_1 \ldots \lambda u_k.(t, new(\lambda u.p)))|\!\}, & \text{otherwise} \end{cases} \\
new(\lambda u.\lambda y.((\lambda x.y)t)) \ =\ \begin{cases} \emptyset, & \text{if } u = x \\ \lambda y.((\lambda x.\lambda u.y)t) & \text{if } u \in fnv(t) \ \wedge\ u \neq x \\ \lambda y.((\lambda x.new(\lambda u.y))t), & \text{otherwise} \end{cases} \\
new(\lambda u.\lambda y.((\lambda x.\lambda w_1 \ldots \lambda w_k.y)t)) \ =\ \begin{cases} \emptyset, & \text{if } u = x \\ \lambda y.((\lambda x.\lambda u.\lambda w_1 \ldots \lambda w_k.y)t), & \text{if } u \in fnv(t) \wedge\ u \neq x \\ \lambda y.((\lambda x.\lambda w_1 \ldots \lambda w_k.new(\lambda u.y))t), & \\ \qquad \text{otherwise} & \end{cases} \\
new(\lambda u.\{\!|tau(p)|\!\}) \ =\ \{\!|tau(new(\lambda u.p))|\!\} \\
new(\lambda u.(p_1 \uplus p_2)) \ =\ new(\lambda u.p_1) \uplus new(\lambda u.p_2)
\end{array}
$$

Figure 5: The concrete definition of *new* over elements $p \in Api_\perp$.

becomes bound, i.e. it involves bound names and/or variables.

The denotational semantics for the applied pi calculus are defined using a semantic function, $\mathcal{S}(\llbracket A \rrbracket)\ \rho\ \phi_{\mathcal{S}} \in Api_{\perp}$, as shown in Figure 6. The multiset,

$$
\begin{array}{lll}
(\mathcal{S}1) & \mathcal{S}(\llbracket A \mid B \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \mathcal{R}(\llbracket \{\!| A |\!\} \uplus \{\!| B |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}} \\
(\mathcal{S}2) & \mathcal{S}(\llbracket \nu n.A \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = new(\lambda n.\mathcal{R}(\llbracket \{\!| A |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}) \\
(\mathcal{S}3) & \mathcal{S}(\llbracket \nu x.A \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = new(\lambda x.\mathcal{R}(\llbracket \{\!| A |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}) \\
(\mathcal{S}4) & \mathcal{S}(\llbracket \{M/x\} \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \{\!| \lambda y.((\lambda x.y)\varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M)) |\!\} \uplus \mathcal{R}(\llbracket \rho \rrbracket)\ \phi_{\mathcal{S}}[x \mapsto \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M)] \\
(\mathcal{S}5) & \mathcal{S}(\llbracket \mathbf{0} \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \emptyset \\
(\mathcal{S}6) & \mathcal{S}(\llbracket !P \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \bigsqcup \mathcal{F} \\
 & \text{where, } \mathcal{F} = \{\{\!| \perp |\!\}, \mathcal{S}(\llbracket \prod_i P[bnv_i(P)/bnv(P)] \rrbracket)\ \rho\ \phi_{\mathcal{S}} \mid i = 0 \ldots \infty\} & \\
 & \text{and, } bnv_i(P) = \{x_i \mid x \in bnv(P)\} & \\
(\mathcal{S}7) & \mathcal{S}(\llbracket \mathit{if}\ M = L\ \mathit{then}\ P\ \mathit{else}\ Q \rrbracket)\ \rho\ \phi_{\mathcal{S}} = & \\
 & \begin{cases} \mathcal{R}(\llbracket \{\!| P |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}, & \text{if } \Sigma \vdash \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L) \\ \mathcal{R}(\llbracket \{\!| Q |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}, & \text{otherwise} \end{cases} & \\
(\mathcal{S}8) & \mathcal{S}(\llbracket M(y).P \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \{\!| in(u, \lambda y.\mathcal{R}(\llbracket \{\!| P |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}) |\!\} \\
 & & \text{where, } \Sigma \vdash \varphi(\phi_{\mathcal{E}}, M) = u \in Pmv \\
(\mathcal{S}9) & \mathcal{S}(\llbracket \overline{M}\langle L \rangle.P \rrbracket)\ \rho\ \phi_{\mathcal{S}} & = \\
 & \biguplus_{M'(z).P' \in \rho} \{\!| tau(\mathcal{R}(\llbracket \{\!| P |\!\} \uplus \rho[P'/M'(z).P'] \rrbracket)\ \phi_{\mathcal{S}}[z \mapsto \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L)]) |\!\} \uplus & \\
 & \{\!| out(u, \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L), \mathcal{R}(\llbracket \{\!| P |\!\} \uplus \rho \rrbracket)\ \phi_{\mathcal{S}}) |\!\} & \\
 & \text{where, } \Sigma \vdash \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M') \in Pmv & \\
 & \text{and, } \Sigma \vdash \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = u \in Pmv & \\
(\mathcal{R}0) & \mathcal{R}(\llbracket \rho \rrbracket)\ \phi_{\mathcal{S}} & = \biguplus_{A \in \rho} \mathcal{S}(\llbracket A \rrbracket)\ (\rho \backslash \{\!| A |\!\})\ \phi_{\mathcal{S}}
\end{array}
$$

Figure 6: The standard denotational semantics of the applied pi calculus.

ρ, is used to hold extended processes composed in parallel with the analysed extended process. For simplicity, the standard singleton, $\{\!| - |\!\} : \mathcal{EP} \to \wp(\mathcal{EP})$, and the multiset union, $\uplus : \wp(\mathcal{EP}) \times \wp(\mathcal{EP}) \to \wp(\mathcal{EP})$, are overloaded from their definitions in (9) and (10) to deal with elements of ρ. The environment, $\phi_{\mathcal{S}} : V \to T$, records any term substitutions that occur in the semantics and the special function, $\varphi_{\mathcal{S}} : (V \to T) \times \mathcal{T} \to T$, returns the semantic value of a term using $\phi_{\mathcal{S}}$:

$$
\varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = \begin{cases} \phi_{\mathcal{S}}(M), & \text{if } M \in \mathcal{V} \wedge M \in dom(\phi_{\mathcal{S}}) \\ M, & \text{if } M \in \mathcal{N} \vee \\ & \quad (M \in \mathcal{V}\ \wedge\ M \notin dom(\phi_{\mathcal{S}})) \\ f(\varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M_1) \ldots \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M_n)), & \\ & \quad \text{if } M = f(M_1, \ldots, M_n) \end{cases}
$$

Intuitively, in rules ($\mathcal{S}2$) and ($\mathcal{S}3$) (which also match the cases of processes, $P, Q \in \mathcal{P}$), interactions are allowed to take place between the extended process with a restricted name (variable) before the effects of that restriction are interpreted from the context's perspective using *new*. Unwanted interactions involving restricted channels will never occur, since bound names (variables) are initially distinct (Property 1) and this distinction is preserved at runtime by Rule ($\mathcal{S}6$). The interpretation of active substitutions in rule ($\mathcal{S}4$) preserves

the structural congruence rule $\{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\}$, since the substitution is recorded in $\phi_{\mathcal{S}}$ for the rest of the interpretation of A. Rule ($\mathcal{S}6$) interprets a replicated process, $!P$, as the least upper bound of the poset, $\mathcal{F}$. This least upper bound is the least fixed point meaning of $!P$. Due to the fact that the semantic domain, $Api_{\perp}$, is infinite, the calculation of this least fixed point may not terminate within finite limits. The rule also uses a labelling mechanism to maintain distinct occurrences of bound variables and names of the spawned processes by subscripting those variables and names with a number signifying each spawned copy. For the case of embedded replications, e.g. $!!\nu u.P$, labelled instances of u resulting from the outside replication still form distinct seeds for instances resulting from the inside replication. For example, the outer labelling will yield $!\nu u_1.P$, $!\nu u_2.P$ etc., whereas the inner labelling will yield $\nu u_{11}.P$, $\nu u_{12}.P$, $\nu u_{21}.P$, $\nu u_{22}.P$, etc. In rule ($\mathcal{S}7$), an *if-then-else* condition is resolved based on the equality of two terms under Σ. No communications are considered in rule ($\mathcal{S}8$) for input actions since these are considered in rule ($\mathcal{S}9$) for output actions. Note that rule ($\mathcal{S}9$) covers general reductions, where the output message is a primitive or a complex term.

Theorem 1 *The interpretation of extended processes in $Api_{\perp}$ is sound and adequate with respect to late transitions in the applied pi calculus (as given in Appendix A).*

Proof sketch. The soundness property relies on the ability of semantic elements to match transitions in the operational model. For example, the reduction rule:

$$\overline{a}\langle x\rangle.P \mid a(x).Q \xrightarrow{\tau} P \mid Q$$

is matched by rule ($\mathcal{S}9$) of Figure 6 by setting $L = z = x$ and $M = M' = a$ as well as noting that the value of $\phi_{\mathcal{S}}$ does not change (hence preserving the current value of x). On the other hand, adequacy requires that the semantic transitions be mapped correctly to the operational model. For example:

$$\begin{array}{l} out(a, \lambda x.\lambda u_1 \ldots \lambda u_k.(t,q)) \in \mathcal{S}([\![A]\!])\ \rho\ \phi_{\mathcal{S}} \ \Rightarrow \\ \exists B, A \in \mathcal{EP}, M \in \mathcal{T} : \\ \nu x.A \xrightarrow{\nu x.\nu\, u_1 \ldots \nu\, u_k.\overline{a}\langle M\rangle} B \ \wedge \ \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = t \ \wedge \ \mathcal{S}([\![B]\!])\ \rho\ \phi_{\mathcal{S}} = q \end{array}$$

The proof of soundness is by induction on semantic rules, whereas the proof of adequacy requires a formal approximation relation, $p \lhd A$, between elements $p \in Api_{\perp}$ and extended processes, $A \in \mathcal{EP}$. Such approximation relations are popular in similar adequacy proofs for the λ-calculus (see for example [28]). □

4 Non-Standard Semantics

The standard meaning of an extended process does not return information on the property we are interested in, i.e. term substitutions. Therefore, to trace

term substitutions during the evolution of extended processes, we define a special environment, $\phi_{\mathcal{E}} : V \rightarrow \wp(T)$, that maps each variable of an extended process to the set of semantic terms that may substitute that variable. Since the non-standard semantics is precise (copies of bound names and variables are always distinct), each variable will be mapped to at most a singleton set per choice of control of flow (on either side of $\uplus$ and $\cup_\phi$). Based on $\phi_{\mathcal{E}}$, a domain, $D_\perp = V \rightarrow \wp(T)$, is constructed, ordered by subset inclusion:

$$\forall \phi_{\mathcal{E}1}, \phi_{\mathcal{E}2} \in D_\perp : \phi_{\mathcal{E}1} \sqsubseteq_{D_\perp} \phi_{\mathcal{E}2} \Leftrightarrow \forall x \in V : \phi_{\mathcal{E}1}(x) \subseteq \phi_{\mathcal{E}2}(x)$$

with the bottom element, $\perp_{D_\perp}$, being the null environment, $\phi_{\mathcal{E}0}$, that maps each variable to the empty set. The union of environments, $\cup_\phi$, is defined as:

$$\forall \phi_{\mathcal{E}1}, \phi_{\mathcal{E}2} \in D_\perp, x \in V : (\phi_{\mathcal{E}1} \cup_\phi \phi_{\mathcal{E}2})(x) = \phi_{\mathcal{E}1}(x) \cup \phi_{\mathcal{E}2}(x)$$

The non-standard semantic domain is formed by pairing $D_\perp$ with the standard semantic domain, $Api_\perp$, resulting in $Api_\perp \times D_\perp$. The bottom element of this domain is the pair $(\perp_{Api_\perp}, \perp_{D_\perp})$.

The non-standard semantics for the applied pi calculus is defined by the semantic function, $\mathcal{E}(\!|A|\!)\ \rho\ \phi_{\mathcal{E}} \in (Api_\perp \times D_\perp)$, as in Figure 7, where $fst(x_1, x_2) = x_1$ and $snd(x_1, x_2) = x_2$.

$(\mathcal{E}1)$	$\mathcal{E}(\!\|A \mid B\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$\mathcal{R}(\!\|\{\!\|A\|\!\} \uplus \{\!\|B\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}$
$(\mathcal{E}2)$	$\mathcal{E}(\!\|\nu n.A\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(new(\lambda n.fst(\mathcal{R}(\!\|\{\!\|A\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}})), snd(\mathcal{R}(\!\|\{\!\|A\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}))$
$(\mathcal{E}3)$	$\mathcal{E}(\!\|\nu x.A\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(new(\lambda x.fst(\mathcal{R}(\!\|\{\!\|A\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}})), snd(\mathcal{R}(\!\|\{\!\|A\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}))$
$(\mathcal{E}4)$	$\mathcal{E}(\!\|\{M/x\}\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(\{\!\|\lambda y.((\lambda x.y)\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M))\|\!\} \uplus fst(\mathcal{R}(\!\|\rho\|\!)\ \phi'_{\mathcal{E}}), snd(\mathcal{R}(\!\|\rho\|\!)\ \phi'_{\mathcal{E}}))$ where, $\phi'_{\mathcal{E}} = \phi_{\mathcal{E}}[x \mapsto \{\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M)\}]$
$(\mathcal{E}5)$	$\mathcal{E}(\!\|\mathbf{0}\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(\emptyset, \phi_{\mathcal{E}})$
$(\mathcal{E}6)$	$\mathcal{E}(\!\|!P\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$\bigsqcup \mathcal{F}$ where, $\mathcal{F} = \{(\perp_{Api_\perp}, \perp_{D_\perp}), \mathcal{E}(\!\|\prod_i P[bnv_i(P)/bnv(P)]\ \|\!)\ \rho\ \phi_{\mathcal{E}} \mid\ i = 0 \ldots \infty\}$ and, $bnv_i(P) = \{x_i \mid x \in bnv(P)\}$
$(\mathcal{E}7)$	$\mathcal{E}(\!\|if\ M = L\ then\ P\ else\ Q\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$\begin{cases} \mathcal{R}(\!\|\{\!\|P\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}, & \text{if } \Sigma \vdash\ \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L) \\ \mathcal{R}(\!\|\{\!\|Q\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}, & \text{otherwise} \end{cases}$
$(\mathcal{E}8)$	$\mathcal{E}(\!\|M(y).P\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(\{\!\|in(u, \lambda y.fst(\mathcal{R}(\!\|\{\!\|P\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}))\|\!\}, \phi_{\mathcal{E}})$ where, $\Sigma \vdash\ \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = u \in Pmv$
$(\mathcal{E}9)$	$\mathcal{E}(\!\|\overline{M}\langle L\rangle.P\|\!)\ \rho\ \phi_{\mathcal{E}}$	$=$	$(((\biguplus_{M'(z).P' \in \rho} \{\!\|tau(fst(\mathcal{R}(\!\|\{\!\|P\|\!\} \uplus \rho[P'/M'(z).P']\|\!)\ \phi_{\mathcal{E}}[z \mapsto \{\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L)\}]))\|\!\}) \uplus \{\!\|out(u, \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L), fst(\mathcal{R}(\!\|\{\!\|P\|\!\} \uplus \rho\|\!)\ \phi_{\mathcal{E}}))\|\!\}), ((\bigcup_{\phi M'(z).P' \in \rho}^{\phi} snd(\mathcal{R}(\!\|\{\!\|P\|\!\} \uplus \rho[P'/M'(z).P']\|\!)\ \phi_{\mathcal{E}}[z \mapsto \{\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L)\}]))\ \cup_\phi\ \phi_{\mathcal{E}}))$ where, $\Sigma \vdash\ \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M') \in Pmv$ and, $\Sigma \vdash\ \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = u \in Pmv$
$(\mathcal{R}0)$	$\mathcal{R}(\!\|\rho\|\!)\ \phi_{\mathcal{E}}$	$=$	$(\biguplus_{A \in \rho} fst(\mathcal{E}(\!\|A\|\!)\ (\rho \backslash \{\!\|A\|\!\})\ \phi_{\mathcal{E}}), \bigcup_{A \in \rho}^{\phi} snd(\mathcal{E}(\!\|A\|\!)\ (\rho \backslash \{\!\|A\|\!\})\ \phi_{\mathcal{E}}))$

Figure 7: The non-standard semantics of the applied pi calculus.

The definition of the function, $\varphi_{\mathcal{E}} : (V \rightarrow \wp(T)) \times \mathcal{T} \rightarrow T$, allows for a term

to be closed as much as possible based on a given $\phi_{\mathcal{E}}$:

$$\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = \begin{cases} t, & \text{if } M \in \mathcal{V} \ \wedge \ \phi_{\mathcal{E}}(M) = \{t\} \\ M, & \text{if } M \in \mathcal{N} \vee \\ & \quad (M \in \mathcal{V} \ \wedge \ \phi_{\mathcal{E}}(M) = \{\}) \\ f(\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M_1) \ldots \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M_n)), & \\ & \text{if } M = f(M_1, \ldots, M_n) \end{cases}$$

The non-standard semantic rules are essentially similar to the standard semantics of the previous section except that the environment holding substitutions, $\phi_{\mathcal{E}}$, is returned as part of the overall meaning. Rules ($\mathcal{E}2$) and ($\mathcal{E}3$) interpret the meaning of a restriction using the *new* operation on the first element of the resulting pair, whereas the second element reflects the $\phi_{\mathcal{E}}$ environment resulting from the scope. This is justified as internal substitutions are preserved by restrictions and Property 1. However, as rule ($\mathcal{E}4$) reveals, systems like $A \stackrel{\texttt{def}}{=} \nu x.\{M/x\}$ will still indicate that a substitution $\phi_{\mathcal{E}}[x \mapsto M]$ is possible. This is reasonable, since it is only to the context of A that such substitutions cannot occur. Furthermore, we know from Property 1 that x will never occur in such a context.

The meaning of a replicated process in rule ($\mathcal{E}6$) is defined as the least upper bound of the poset, $\mathcal{F}$, of non-standard semantic elements. Since the non-standard semantic domain, $Api_{\perp} \times D_{\perp}$, is infinite, it may not be possible to calculate this least fixed point within finite limits. Rules ($\mathcal{E}8$) and ($\mathcal{E}9$) deal with the cases of input and output actions, respectively. Communications are dealt with in rule ($\mathcal{E}9$) for output actions as usual and therefore, $\phi_{\mathcal{E}}$ remains unchanged in rule ($\mathcal{E}8$) for input actions. Finally, rule ($\mathcal{R}0$) is used to interpret the contents of the ρ multiset using $\uplus$ and $\cup_{\phi}$.

The correctness requirement for the non-standard semantics with respect to the standard semantics is expressed as follows.

Theorem 2 $\forall A \in \mathcal{EP} : (\mathcal{S}(\llbracket A \rrbracket) \ \rho \ \phi_{\mathcal{S}} = p) \ \wedge \ (\mathcal{E}(\llbracket A \rrbracket) \ \rho \ \phi_{\mathcal{E}} = (p', \phi'_{\mathcal{E}})) \ \Rightarrow$
$(p' = p) \ \wedge \ (\exists t \in T, x, y \in V : t \in \phi'_{\mathcal{E}}(x) \ \Rightarrow \ \lambda y.((\lambda x.y)t) \in p)$

Proof sketch. The correctness requirement can be shown by induction over the rules of the non-standard and standard semantics. □

The theorem gives two correctness requirements: the first is that the standard part of a non-standard interpretation must be the same as the result of the standard interpretation. The second requirement is that for any substitutions that appear in the result of the standard interpretation, then those substitutions must be captured by the final non-standard environment, $\phi'_{\mathcal{E}}$.

5 Abstract Semantics

In order to limit the size of the semantic domain to a finite limit, we adopt the abstraction of [10, 12], which limits the number of copies of bound names and

variables and the depth of complex terms. For this purpose, we first introduce a finite predomain of tags, Tag, ranged over by $t, \dot{t}, \ddot{t}$, where t is the tag of any term, $\dot{t}$ is the tag of a name or a variable and $\ddot{t}$ is the tag of a function application. Using Tag, we tag L in occurrences of $\overline{M}\langle L\rangle.P$ and $\{L/x\}$ in the syntax. The following functions are also defined over tags and processes:

$$value\_of \quad : \quad \wp(Tag) \rightarrow \wp(\mathcal{T}) \tag{12}$$

$$tags\_of \quad : \quad \mathcal{P} \rightarrow \wp(Tag) \tag{13}$$

such that we have, $value\_of(\{t_1, \ldots, t_n\}) = \{M_1, \ldots, M_n\}$ returns a set of tagged terms corresponding to a set of tags. For example, $value\_of(\{t_1, t_2\}) = \{M, N\}$ in $A \stackrel{\texttt{def}}{=} \overline{a}\langle M^{t_1}\rangle.P \mid \{N^{t_2}/x\}$. On the other hand, $tags\_of(P) = \{t_1, \ldots, t_n\}$ returns the set of tags used in a process. For example, $tags\_of(A) = \{t_1, t_2\} \cup tags\_of(P)$.

We now introduce the $\alpha_{k,k'}$ abstraction function, which keeps to a finite level, the number of copies of primitive terms and tags.

Definition 1 *Define* $\alpha_{k,k'} : \mathbb{N} \times \mathbb{N} \times (Pmv + Tag) \rightarrow (Pmv^\sharp + Tag^\sharp)$*:*
$\forall \mathit{ut} \in (Pmv + Tag), i, k, k' \in \mathbb{N} : \alpha_{k,k'}(\mathit{ut}) =$

$$\begin{cases} \dot{t}_k, & \text{if } \mathit{ut} = \dot{t}_i \in Tag \text{ and } i > k \\ \ddot{t}_{k'}, & \text{if } \mathit{ut} = \ddot{t}_i \in Tag \text{ and } i > k' \\ u_k, & \text{if } \mathit{ut} = u_i \in Pmv \text{ and } i > k \\ \mathit{ut}, & \text{otherwise} \end{cases}$$

For example, we have that $\alpha_{2,3}(\dot{t}_{10}) = \dot{t}_2$, $\alpha_{2,3}(\ddot{t}_5) = \dot{t}_3$ and $\alpha_{2,3}(x_2) = x_2$. The resulting abstract predomains, $Pmv^\sharp$ and $Tag^\sharp$, are defined as $Pmv^\sharp = Pmv \backslash \{u_j \mid j > k\}$ and $Tag^\sharp = Tag \backslash (\{\dot{t}_j \mid j > k\} \cup \{\ddot{t}_i \mid i > k'\})$. Informally, k constrains the number of copies of primitive terms and their tags, whereas k' constrains the number of copies of tags of function applications. In effect, constraining the tags of primitive terms implies limiting the copies of bound names and variables carrying the tags, whereas constraining the number of tags of function applications means limiting the depth of the resulting data structures. For example, in the process, $!\nu n.\overline{a}\langle n^{\dot{t}}\rangle \mid !a(x)$, it is possible to spawn infinite copies of each replication, $\nu n_1.\overline{a}\langle n_1^{\dot{t}_1}\rangle \mid a(x_1) \mid \nu n_2.\overline{a}\langle n_2^{\dot{t}_2}\rangle \mid a(x_2) \ldots$, making it clear that $\dot{t}$ is an indicator to the number of copies n has after spawning each process. On the other hand, the process, $!a(x).\overline{a}\langle \texttt{enc}(x, k)^{\ddot{t}}\rangle \mid \overline{a}\langle b\rangle$, which spawns $a(x_1).\overline{a}\langle \texttt{enc}(x_1, k)^{\ddot{t}_1}\rangle \mid a(x_2).\overline{a}\langle \texttt{enc}(x_2, k)^{\ddot{t}_2}\rangle \mid \overline{a}\langle b\rangle \ldots$, demonstrates the role of $\ddot{t}$ as a counter of the number of times the encryption is applied to b. The choice between a uniform ($k = k' = 1$) and a non-uniform ($k > 1$ or $k' > 1$) analysis is a matter of trade-off between precision and cost.

Using $\alpha_{k,k'}$, we define the abstract environment, $\phi_{\mathcal{A}} : V^\sharp \rightarrow \wp(Tag^\sharp)$, which maps each abstract bound variable of the analysed extended process to a set of abstract tags, representing terms that could substitute that variable. An abstract domain, $D_\perp^\sharp = V^\sharp \rightarrow \wp(Tag^\sharp)$, is formed ordered by subset inclusion, as follows:

$$\forall \phi_{\mathcal{A}1}, \phi_{\mathcal{A}2} \in D^\sharp_\bot, x \in V^\sharp : \phi_{\mathcal{A}1} \sqsubseteq_{D^\sharp_\bot} \phi_{\mathcal{A}2} \Leftrightarrow \phi_{\mathcal{A}1}(x) \subseteq \phi_{\mathcal{A}2}(x)$$

where, $\bot_{D^\sharp_\bot} = \phi_{\mathcal{A}0}$. Taking $D^\sharp_\bot$ as the abstract semantic domain, we define the abstract semantic function, $\mathcal{A}(\![A]\!)\ \rho\ \phi_{\mathcal{A}} \in D^\sharp_\bot$, as in Figure 8.

$$
\begin{array}{lll}
(\mathcal{A}1) & \mathcal{A}(\![A \mid B]\!)\ \rho\ \phi_{\mathcal{A}} & = \mathcal{R}(\![\{\!|A|\!\} \uplus \{\!|B|\!\} \uplus \rho]\!)\ \phi_{\mathcal{A}} \\
(\mathcal{A}2) & \mathcal{A}(\![\nu n.A]\!)\ \rho\ \phi_{\mathcal{A}} & = \mathcal{R}(\![\{\!|A|\!\} \uplus \rho]\!)\ \phi_{\mathcal{A}} \\
(\mathcal{A}3) & \mathcal{A}(\![\nu x.A]\!)\ \rho\ \phi_{\mathcal{A}} & = \mathcal{R}(\![\{\!|A|\!\} \uplus \rho]\!)\ \phi_{\mathcal{A}} \\
(\mathcal{A}4) & \mathcal{A}(\![\{M^t/x\}]\!)\ \rho\ \phi_{\mathcal{A}} & = \mathcal{R}(\![\rho]\!)\ \phi_{\mathcal{A}}[x \mapsto \phi_{\mathcal{A}}(x) \cup \{t\}] \\
(\mathcal{A}5) & \mathcal{A}(\![\mathbf{0}]\!)\ \rho\ \phi_{\mathcal{A}} & = \phi_{\mathcal{A}} \\
(\mathcal{A}6) & \mathcal{A}(\![!P]\!)\ \rho\ \phi_{\mathcal{A}} & = \bigsqcup \mathcal{F} \\
 & \text{where, } \mathcal{F} = \{\bot_{D^\sharp_\bot}, \mathcal{A}(\![\prod_i ren(P,i)\ ]\!)\ \rho\ \phi_{\mathcal{A}} \mid\ i = 0 \ldots \infty\} & \\
 & \text{and, } ren(P,i) = fold\ sub_i\ (fold\ sub_i\ P\ bnv(P))\ tags\_of(P) & \\
 & \text{and, } fold\ f\ e\ \{x_1, \ldots, x_n\} = f(x_n, \ldots, f(x_1, e)\ldots) & \\
 & \text{and, } sub_i\ x\ y = y[\alpha_{k,k'}(x_i)/x] & \\
(\mathcal{A}7) & \mathcal{A}(\![\mathit{if}\ M = L\ \mathit{then}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{A}} = & \\
 & \begin{cases} \mathcal{R}(\![\{\!|P|\!\} \uplus \rho]\!)\ \phi_{\mathcal{A}}, & \text{if } \exists M' \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M), L' \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L) :\ \Sigma \vdash\ M' =^\sharp L' \\ \mathcal{R}(\![\{\!|Q|\!\} \uplus \rho]\!)\ \phi_{\mathcal{A}}, & \text{otherwise} \end{cases} & \\
(\mathcal{A}8) & \mathcal{A}(\![\underline{M}(y).P]\!)\ \rho\ \phi_{\mathcal{A}} & = \phi_{\mathcal{A}} \\
(\mathcal{A}9) & \mathcal{A}(\![\overline{M}\langle L^t\rangle.P]\!)\ \rho\ \phi_{\mathcal{A}} & = (\bigcup_{\phi\ M'(z).P' \in \rho} \mathcal{R}(\![\{\!|P|\!\} \uplus \rho[P'/M'(z).P']]\!)\ \phi'_{\mathcal{A}})\ \cup_\phi\ \phi_{\mathcal{A}} \\
 & \text{where, } \exists M'' \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M), M''' \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M') :\ \Sigma \vdash M'' =^\sharp M''' \in Pmv & \\
 & \text{and, } \phi'_{\mathcal{A}} = \phi_{\mathcal{A}}[z \mapsto\ \phi_{\mathcal{A}}(z) \cup \{t\}] & \\
(\mathcal{R}0) & \mathcal{R}(\![\rho]\!)\ \phi_{\mathcal{A}} & = \bigcup_{\phi\ A \in \rho} \mathcal{A}(\![A]\!)\ (\rho \backslash \{\!|A|\!\})\ \phi_{\mathcal{A}}
\end{array}
$$

Figure 8: The abstract semantics of the applied pi calculus.

The special function, $\varphi_{\mathcal{A}} : (V^\sharp \to \wp(Tag^\sharp)) \times \mathcal{T} \to \wp(T)$, returns a set of terms corresponding to a term, M, given tag substitutions captured by $\phi_{\mathcal{A}}$:

$$\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M) = \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M[\alpha_{k,k'}(x)/x][\alpha_{k,k'}(n)/n])_{\{\}}$$
where,
$$\varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M)_s = \mathit{if}\ M \in s\ \mathit{then}\ \{\}\ \mathit{else}$$
$$\begin{cases} \bigcup_{L \in value\_of(\phi_{\mathcal{A}}(M))} \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, L)_{s \cup \{M\}}, \\ \text{if } M \in V^\sharp\ \wedge\ \phi_{\mathcal{A}}(M) \neq \{M\} \\ \{M\}, \\ \text{if } M \in N^\sharp\ \vee\ (M \in V^\sharp\ \wedge\ \phi_{\mathcal{A}}(M) = \{\}) \\ \{f(M'_1, \ldots, M'_n) \mid M'_1 \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M_1)_{s \cup \{M\}}, \\ \ldots, M'_n \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M_n)_{s \cup \{M\}}\}, \\ \text{if } M = f(M_1, \ldots, M_n) \end{cases}$$

In the above definition of $\varphi'_{\mathcal{A}}$, the set s is required to prevent the computation from looping infinitely, in case cyclic terms are encountered, for example, $\phi_{\mathcal{A}}[x \to \{t\}]$ and $value\_of(t) = \texttt{encrypt}(x, k)$.

Rules ($\mathcal{A}2$) and ($\mathcal{A}3$) return the $\phi_{\mathcal{A}}$ environment resulting from the evolution of an extended process under a top-level restriction. Property 1 plays an important role in disallowing incorrect behaviour between the scope and the

rest of elements in ρ. Rule ($\mathcal{A}4$) deals with floating substitutions by updating $\phi_{\mathcal{A}}$ with the tag of the substituted term. The rule for replication, ($\mathcal{A}6$), defines the least upper bound of the poset, $\mathcal{F}$, as the least fixed point of the meaning of a replicated process while labelling bound names and variables as well as tags of the spawned copies. This rule however is different from its concrete versions of Figures 6 and 7 in that it abstracts every bound primitive term and tag belonging to the spawned processes using the abstraction, $\alpha_{k,k'}$. This maintains that the number of resulting copies of those primitive terms and tags is kept finite. The following theorem states that the calculation of the least fixed point in abstract semantics is guaranteed to terminate due to the finite nature of $D^{\sharp}_{\perp}$.

Theorem 3 *The calculation of rule* ($\mathcal{A}6$) *terminates.*

Proof sketch. To prove the termination property, it is necessary to satisfy the following requirements. First, the semantic domain must be finite. This is satisfied by the definition of $D^{\sharp}_{\perp}$. Second, the size of the Σ must be finite, i.e. there must be a finite number of equations relating terms. Finally, we need to prove the monotonicity of $\mathcal{A}(\!|\prod_i P|\!)\ \rho\ \phi_{\mathcal{A}}$, i.e. $\mathcal{A}(\!|\prod_i P|\!)\ \rho\ \phi_{\mathcal{A}} \sqsubseteq_{D^{\sharp}_{\perp}} \mathcal{A}(\!|\prod_{i+1} P|\!)\ \rho\ \phi_{\mathcal{A}}$. To prove this, we simplify the inequality into $\mathcal{A}(\!|Q|\!)\ \rho\ \phi_{\mathcal{A}} \sqsubseteq_{D^{\sharp}_{\perp}} \mathcal{A}(\!|Q \mid P|\!)\ \rho\ \phi_{\mathcal{A}}$, where $Q = \prod_i P$. This is further simplified to become $\mathcal{A}(\!|Q|\!)\ \rho\ \phi_{\mathcal{A}} \sqsubseteq_{D^{\sharp}_{\perp}} \mathcal{A}(\!|Q|\!)\ \rho'\ \phi_{\mathcal{A}}$, where $\rho' = \rho \uplus_{\rho} \{\!|P|\!\}_{\rho}$. This can be proven by induction over P. In particular, the most interesting cases are rules ($\mathcal{A}4$) and ($\mathcal{A}9$), where $\phi_{\mathcal{A}}$ changes. For example, in rule ($\mathcal{A}9$), we have that since $\rho \subseteq \rho'$, then $M'(y).P' \in \rho \Rightarrow M'(y).P' \in \rho'$. From this we can conclude that $\mathcal{A}(\!|Q|\!)\ \rho\ \phi_{\mathcal{A}} \sqsubseteq_{D^{\sharp}_{\perp}} \mathcal{A}(\!|Q|\!)\ \rho'\ \phi_{\mathcal{A}}$, since the environment resulting from $\mathcal{A}(\!|Q|\!)\ \rho\ \phi_{\mathcal{A}}$ will necessarily be a subset of the environment resulting from $\mathcal{A}(\!|Q|\!)\ \rho'\ \phi_{\mathcal{A}}$. Informally, this means that the larger system will always induce more term substitutions than the smaller system. □

The rule for conditional processes, ($\mathcal{A}7$), identifies two terms whenever it finds terms in their $\phi_{\mathcal{A}}$ tag values that are *abstract equal* under Σ, in other words, $\Sigma \vdash M' =^{\sharp} L'$. The definition of $=^{\sharp}$ is given as:

$$
\begin{array}{l}
\Sigma \vdash M' =^{\sharp} L' \ \Leftrightarrow \\
\exists M, L:\ (\Sigma \vdash M = L) \wedge \\
(M' = M[\alpha_{k,k'}(x)/x]_{x \in bnv(M)}) \wedge \\
(L' = L[\alpha_{k,k'}(x)]_{x \in bnv(L)})
\end{array}
$$

It is worth noting that rule ($\mathcal{A}7$) introduces further approximation when considering that each of the compared terms may have a non-singleton set of values in $\phi_{\mathcal{A}}$.

The rule for input actions, ($\mathcal{A}8$), returns the $\phi_{\mathcal{A}}$ environment unchanged since communications are dealt with in the next rule, ($\mathcal{A}9$), which deals with output actions. The $\phi_{\mathcal{A}}$ environment is updated with appropriate tags for each communication using $\alpha_{k,k'}$. Finally, rule ($\mathcal{R}0$) unifies all the interpretations of extended processes composed in ρ, using $\cup_{\phi}$.

We can state the safety of the abstract semantics by the following theorem.

Theorem 4 $\forall A, \rho, \phi_{\mathcal{E}}, \phi_{\mathcal{A}}, k, k', \mathcal{E}([\![A]\!])\rho\phi_{\mathcal{E}} = (p, \phi'_{\mathcal{E}}), \mathcal{A}([\![A]\!])\ \rho\ \phi_{\mathcal{A}} = \phi'_{\mathcal{A}}$:
$(\exists M, M' \in \mathcal{T}, x \in \mathcal{V} : \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) \in \phi_{\mathcal{E}}(x)\ \Rightarrow$
$\exists t \in \phi_{\mathcal{A}}(\alpha_{k,k'}(x)) : value\_of(\{t\}) = \{fold\ sub_{k,k'}\ M'\ bnv(M')\} \wedge \Sigma \vdash M = M')$
$\Rightarrow$
$(\exists M, M' \in \mathcal{T}, x \in \mathcal{V} : \varphi_{\mathcal{E}}(\phi'_{\mathcal{E}}, M) \in \phi'_{\mathcal{E}}(x)\ \Rightarrow$
$\exists t \in \phi'_{\mathcal{A}}(\alpha_{k,k'}(x)) : value\_of(\{t\}) = \{fold\ sub_{k,k'}\ M'\ bnv(M')\} \wedge \Sigma \vdash M = M')$
where,
$fold\ f\ e\ \{x_1, \ldots, x_n\} = f(x_n, \ldots, f(x_1, e)\ldots)$
and, $sub_{k,k'}\ x\ y = y[\alpha_{k,k'}(x)/x]$

Proof sketch. The proof is by induction over the structure of the abstract semantics and relies on the safety of the $\cup_\phi$ operation. □

The theorem states that for any term, M, captured in the non-standard semantics by including its $\varphi_{\mathcal{E}}(\phi'_{\mathcal{E}}, M)$ value in the value of a variable, $\phi'_{\mathcal{E}}(x)$, then that will correspond to capturing a tag, t, in the abstract semantics, by $\phi'_{\mathcal{A}}(\alpha_{k,k'}(x))$. The appropriateness of t is expressed by the ability to obtain an abstract form, $fold\ sub_{k,k'}\ M'\ bnv(M'))$, of a term, M', that is equivalent under Σ to the concrete term, M, by evaluating t using *value_of*. More concisely, every concrete term, M, captured in the non-standard semantics is captured as the corresponding abstract tag, t, in the abstract semantics.

6 Security Properties

The two main security properties we consider in this section, *information leakage* and *authenticity breach*, were essentially introduced earlier in [9, 10, 12]. In what follows, we adapt these properties for the results of the abstract semantics of the previous section.

Given, $S = (SL, \sqsubseteq_S, \sqcap_S, \sqcup_S, \bot_S, \top_S)$ and $A = (AL, \sqsubseteq_A, \sqcap_A, \sqcup_A, \bot_A, \top_A)$ as finite lattices of secrecy and trust levels, respectively, and $\xi_S : (\mathcal{N} \cup \mathcal{V}) \to S$ and $\xi_A : (\mathcal{N} \cup \mathcal{V}) \to A$ as classification environments from names and variables to secrecy and trust levels, respectively (where the null environments are defined as $\forall u \in (\mathcal{N} \cup \mathcal{V}) : \xi_{S0}(u) = \bot_S\ \wedge\ \xi_{A0}(u) = \bot_A$, indicating that the secrecy and trust levels of free names are the bottom elements, which is the safest assumption one could assume). Now we can define the following properties.

Definition 2 (Information leakage)
A name, a, is leaked in an extended process, A, if given $\phi_{\mathcal{A}} = \mathcal{A}([\![A]\!])\ \rho_0\phi_{\mathcal{A}0}$ *and a classification environment,* ξ_S*, then the following holds:* $\exists y \in dom(\phi_{\mathcal{A}}), u \in (\mathcal{N} \cup \mathcal{V}), M \in value\_of(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, y)) : \xi_S(y) \sqsubseteq_S \xi_S(a)\ \wedge\ \Sigma \vdash M = u$

Definition 3 (Authenticity breach) *The authenticity requirement of a variable, y, is breached within an extended process, A, if given the abstract interpretation,* $\phi_{\mathcal{A}} = \mathcal{A}([\![A]\!])\ \rho_0\phi_{\mathcal{A}0}$ *and a classification environment,* ξ_A*, then the following holds:* $\exists y \in dom(\phi_{\mathcal{A}}), u \in (\mathcal{N} \cup \mathcal{V}), M \in value\_of(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, y)) : \xi_A(a) \sqsubseteq_A \xi_A(y)\ \wedge\ \Sigma \vdash M = u$

Information leaks occur whenever data classified at high secrecy levels are captured by variables classified at lower secrecy levels, whereas authenticity breaches have an opposite direction of concern compared to information leaks: a variable with high trust level breaches authenticity whenever it manages to capture a data item with low trust level. Note, that the properties capture the secrecy and trust levels only of terms that are primitive terms under the equational theory of Σ. This stems from the realisation that only primitive terms may be thought of as sensitive data that need to be secured and monitored. Complex terms have the sole purpose of securely carrying the sensitive information across the network, hence, a complex term is useless to a recipient that is unable to destruct it and extract its sensitive data.

7 Example: The Diffie-Hellman Protocol

We consider here a variant of the Diffie-Hellman key exchange protocol [20] to demonstrate the use of equations like $f(x, g(y)) = f(y, g(x))$. The protocol consists of the following exchange of messages between an initiator, agent A, and a responder, agent B, aimed at establishing a session key, K:

Message 1	$A \to B:$	$\{g(N)\}_K$	on c_B
Message 2	$B \to A:$	$\{g(N')\}_K$	on c_A
Message 3	$A \to B:$	$\{M\}_{K_{AB}}$	on c_B
Message 4	$B \to A:$	$\{M'\}_{K_{AB}}$	on c_A
Message 5	$A \to I:$	K	on c_I

where K is a long-term secret key shared between A and B. The two initial messages (1–2) involve sending the terms, $g(x) = l^x \texttt{ mod } p$, where x is instantiated by fresh nonces, N and N', p is a reasonably large prime number and l is a number such that $l < p$. Both p and l are public. The session key is then computed by both the initiator and the responder as $K_{AB} = f(N, g(N')) = f(N', g(N))$, where $f(y, z) = z^y \texttt{ mod } p$. The next two messages (3–4), are used to exchange sensitive data, M and M', encrypted under the session key, K_{AB}. Finally, in message (5), A performs a faulty action by releasing the long-term secret key shared with B to the intruder (the public network). Our main concern is to verify the secrecy and authenticity of messages M, M' and whether the release of K will affect their properties in subsequent runs of the protocol.

The specification of the protocol in the applied pi language is given in Figure 9 (for clarity, we have not tagged any terms). In this specification, we have used non-recursive definitions *Init*, *Resp* and I to describe the behaviours of the protocol initiator, responder and intruder. Here, X is an agent variable representing the identity of the initiator and Y is an agent variable representing the identity of the responder. Both X and Y may be substituted by names of honest agents *Alice*, *Bob* (but not I). Hence, $Init\{A/X\}\{B/Y\}$ indicates that the initiator is agent A and it is initiating the protocol to agent B and $Resp\{A/X\}\{B/Y\}$ to indicate that the responder is agent B and it is expected to respond to A. The different combinations of *Init*/*Resp* and their substitutions

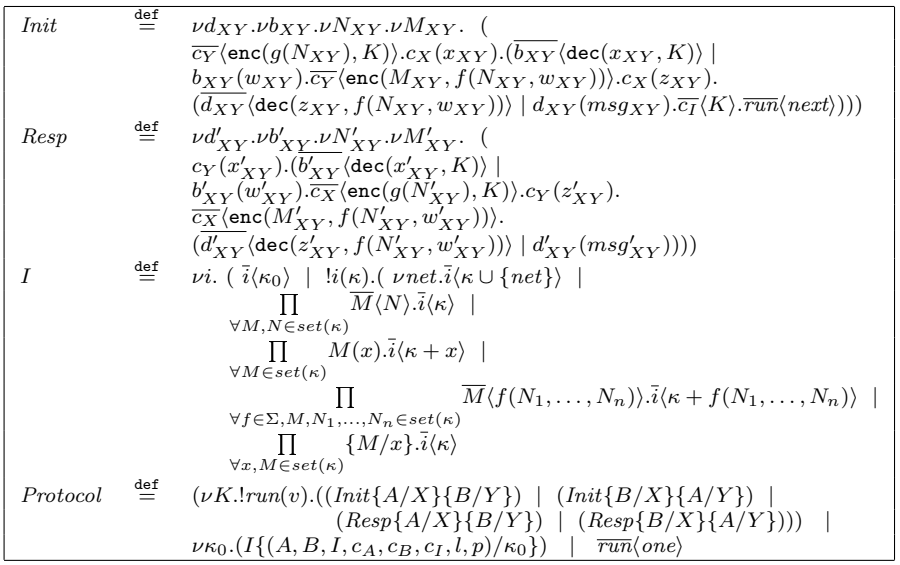

Figure 9: Specification of the Diffie-Hellman protocol including the intruder, I.

in the definition of the *Protocol* do not include the cases where the intruder is the initiator or the responder. This is due to the fact that such behaviours are possible to simulate within the Dolev-Yao definition of I (malicious behaviour always subsumes honest one).

The definition of the intruder, I, itself is an implementation of the Dolev-Yao model [21], modified to include general functions (instead of cryptographic functions only). The initial knowledge of I is given as a tuple, $\kappa_0 = (M_1, \ldots, M_n)$, and $\mathcal{K}(\kappa_0) = \{M_1, \ldots, M_n\}$ is the underlying set. The substitution applied to I in *Protocol* involving κ_0 is the instantiation of this knowledge. Furthermore, the intruder is capable of building up it knowledge by creating new names, *net*, and learning those names, as well as performing input actions over any of the names available in its current knowledge in order to spy on the context. In between, the intruder can also apply any function or perform any substitution over the terms it currently knows about and learn the resulting term. Finally, the intruder also attempts to constantly output all its knowledge to the context.

In order to analyse the protocol, we apply $\mathcal{A}[\![\mathit{Protocol}]\!]\ \{\!|\ |\!\}\ \phi_{\mathcal{A}0}$ with $\alpha_{2,2}$, which enables us to monitor two runs of the protocol (non-uniform analysis). The results are shown in Figure 10 for some of the values of the final $\phi_{\mathcal{A}}$ after the application of the *value_of* function to simplify the figure. We explain these results for two runs initiated by A to B. In the first run, the intruder, I, was unable to capture any of the sensitive messages, M_{AB1}, M'_{AB1}, since I is unable to decipher $\mathtt{enc}(g(N_{AB1}), K)$ or $\mathtt{enc}(g(N'_{AB1}), K)$ as K is only leaked at the end of the first run triggered by $\overline{run}\langle one\rangle$. Furthermore, since I now has K, it is capable in the second run to obtain messages, M_{AB2}, M'_{AB2}. This is due

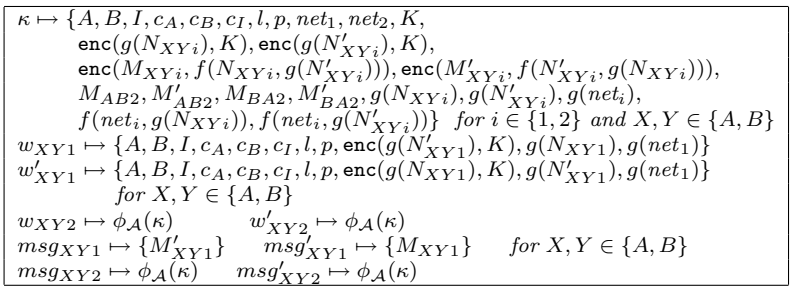

Figure 10: Results of the non-uniform analysis of the Diffie-Hellman protocol.

to the fact that the intruder obtains the session key, $f(net_2, g(N_{AB2})) = K_{IA}$ shared with A, and $f(net_2, g(N'_{AB2})) = K_{IB}$ shared with B. Hence, A will compute $f(N_{AB2}, g(net_2)) = K_{AI}$, and B will compute $f(N'_{AB2}, g(net_2)) = K_{BI}$. Furthermore, by the equations, $\Sigma \vdash K_{IA} = K_{AI}$ and $\Sigma \vdash K_{IB} = K_{BI}$, we have that I now has shared keys with both A and B and therefore, it is capable of acting in the role of the man in the middle.

We find that these results indicate information leakage and authenticity breaches for messages, M_{XY2}, but not for messages M_{XY1}. For example, setting $\xi_S(\kappa) = \bot_S$, $\xi_S(M_{AB2}) = \top_S$, we have then an instance of Property 2 and message M_{AB2} is leaked to I. Similarly, setting $\xi_A(net_2) = \bot_A$ and $\xi_A(msg_{AB2}) = \top_S$, we find that Property 3 is satisfied and the authenticity of variable msg_{AB2} is breached. However, following the same argument, both the secrecy of M_{AB1} and the authenticity of msg_{AB1} are preserved.

Performing a uniform analysis with $\alpha_{1,1}$ would yield results that are less precise, as shown in Figure 11. In these results, messages M_{AB1}, M'_{AB1} are

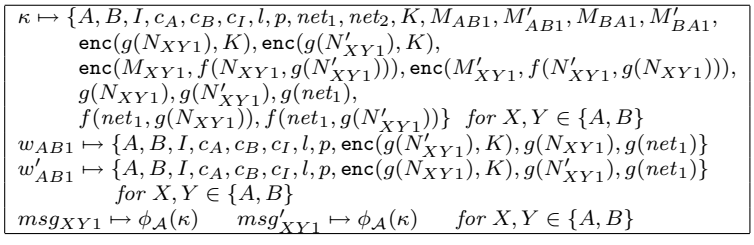

Figure 11: Results of the uniform analysis of the Diffie-Hellman protocol.

captured by the intruder, which is due to the uniformity of the $\alpha_{1,1}$ abstraction, which renders M_{AB2}, M'_{AB2} and M_{AB1}, M'_{AB1} the same. Similarly, variables msg_{AB1}, msg'_{AB1} and msg_{AB2}, msg'_{AB2} are indistinguishable. Consequently, neither the secrecy of messages M_{AB1}, M'_{AB1} nor the authenticity of variables msg_{AB1}, msg'_{AB1} can then be proven.

8 Conclusion and Future Work

We have presented a non-uniform static analysis of the applied pi calculus, which captures the essential property of term substitutions. The analysis follows a denotational approach by first developing a domain-theoretic model of the language, correctly extending the model with the property of interest, and finally, introducing a safe abstraction to ensure termination.

There are several directions for extending the current work. We plan to apply the analysis to more realistic protocols with non-free term algebra and that possibly adopt as their core the Diffie-Hellman protocol. Also, it would be interesting to capture other security properties, e.g. *message deniability* [30], where an agent is incapable of proving to a third party that a message was sent by another agent, even if itself is sure about the message's authenticity. Hence, for example, in the analysis of the Diffie-Hellman protocol, one may be able to prove the deniability of message, $\mathtt{enc}(M_{AB1}, f(N_{AB1}, g(N'_{AB1})))$, since M_{AB1} is obtained by B and B has the term $f(N'_{AB1}, g(N_{AB1}))$, where $\Sigma \vdash f(N'_{AB1}, g(N_{AB1})) = f(N_{AB1}, g(N'_{AB1}))$. Therefore, B is capable of constructing $\mathtt{enc}(M_{AB1}, f(N_{AB1}, g(N'_{AB1})))$ to a third party and consequently, it cannot prove that A was the source of the message.

References

[1] Martín Abadi. Secrecy by typing in security protocols. In Martín Abadi and Takayasu Ito, editors, *Proceedings of the* 3rd *International Symposium on Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 611–638, Sendai, Japan, September 1997. Springer Verlag.

[2] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proceedings of the* 29th *Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 33–44, Portland, USA, January 2002. ACM Press.

[3] Martín Abadi and Bruno Blanchet. Computer-assisted verification of a protocol for certified email. In Radhia Cousot, editor, *Proceedings of the* 10th *International Symposium in Static Analysis*, volume 2694 of *Lecture Notes in Computer Science*, pages 316–335, San Diego, California, U.S.A., June 2003. Springer Verlag.

[4] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. In David Schmidt, editor, *Programming Languages and Systems: Proceedings of the* 13th *European Symposium on Programming*, volume 2986 of *Lecture Notes in Computer Science*, pages 340–354, Barcelona, Spain, March 2004. Springer Verlag.

[5] Martín Abadi and Cédric Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the* 28th *ACM Symposium on Principles*

of Programming Languages, pages 104–115, London, UK, January 2001. ACM Press.

[6] Martín Abadi and Andrew Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the* 4th *ACM Conference on Computer and Communications Security*, pages 36–47, Zurich, Switzerland, April 1997. ACM Press.

[7] Samson Abramsky. A domain equation for bisimulation. *Information and Computation*, 92(2):161–218, June 1991.

[8] Roberto Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In Catuscia Palamidessi, editor, *Proceedings of the* 11th *International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 380–394, Pennsylvania, USA, August 2000. Springer Verlag.

[9] B. Aziz. *A Static Analysis Framework for Security Properties in Mobile and Cryptographic Systems*. PhD thesis, School of Computing, Dublin City University, Dublin, Ireland, 2003.

[10] B. Aziz, G.W. Hamilton, and D. Gray. A denotational approach to the static analysis of cryptographic processes. In *Proceedings of International Workshop on Software Verification and Validation*, volume 118, pages 19–36, Mumbai, India, December 2003. Electronic Notes in Theoretical Computer Science.

[11] Benjamin Aziz and Geoff Hamilton. A privacy analysis for the π-calculus: The denotational approach. In *Proceedings of the* 2nd *Workshop on the Specification, Analysis and Validation for Emerging Technologies*, number 94 in Datalogiske Skrifter, Copenhagen, Denmark, July 2002. Roskilde University.

[12] Benjamin Aziz, Geoff Hamilton, and David Gray. A static analysis of cryptographic processes: The denotational approach. *Journal of Logic and Algebraic Programming*, 64(2):285–320, August 2005.

[13] Bruno Blanchet. From secrecy to authenticity in security protocols. In Manuel V. Hermenegildo and German Puebla, editors, *Proceedings of the* 9th *International Symposium in Static Analysis*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359, Madrid, Spain, September 2002. Springer Verlag.

[14] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Control flow analysis for the π-calculus. In *Proceedings of the* 9th *Conference on Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science*, pages 84–98, Nice, France, September 1998. Springer Verlag.

[15] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis of processes for no read-up and no write-down. In *Proceedings of the Conference on Foundations of Software Science and Computation Structures*, volume 1578 of *Lecture Notes in Computer Science*, pages 120–134, Lisbon, Portugal, March 1999. Springer Verlag.

[16] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for secrecy and non-interference in networks of processes. In *Proceedings of the* 6^{th} *International Conference in Parallel Computing Technologies*, volume 2127 of *Lecture Notes in Computer Science*, pages 27–41, Novosibirsk, Russia, September 2001. Springer Verlag.

[17] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 168(1):68–92, July 2001.

[18] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Proceedings of the* 28^{th} *International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681, Crete, Greece, July 2001. Springer Verlag.

[19] Michele Boreale and Maria Grazia Buscemi. Experimenting with sta: A tool for automatic analysis of security protocols. In *Proceedings of the ACM Symposium on Applied Computing*, pages 281–285, Madrid, Spain, March 2002. ACM Press.

[20] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[21] Danny Dolev and A. Yao. On the security of public key protocols. In *Proceedings of the* 22^{nd} *Annual Symposium on Foundations of Computer Science*, pages 350–357, October 1981.

[22] Cédric Fournet and Martín Abadi. Hiding names: Private authentication in the applied pi calculus. In Mitsuhiro Okada, Benjamin C. Pierce, Andre Scedrov, Hideyuki Tokuda, and Akinori Yonezawa, editors, *Proceedings of the Mext-NSF-JSPS International Symposium International Symposium in Software Security – Theories and Systems*, volume 2609 of *Lecture Notes in Computer Science*, pages 317–338, Tokyo, Japan, November 2002. Springer Verlag.

[23] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proceedings of the FLOC 1999 Formal Methods and Security Protocols Workshop*, pages 21–51, Trento, Italy, July 1999.

[24] Hugo Krawczyk. Skeme: A versatile secure key exchange mechanism for internet. In *Proceedings of the Internet Society Symposium on Network*

and Distributed Systems Security, San Diego, C.A., USA, February 1996. Internet Society.

[25] Fabio Martinelli. Symbolic partial model checking for security analysis. In *Proceedings of the* 2nd *Workshop on the Specification, Analysis and Validation for Emerging Technologies*, Copenhagen, Denmark, July 2002.

[26] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes (parts I & II). *Information and Computation*, 100(1):1–77, September 1992.

[27] David Monniaux. Abstracting cryptographic protocols with tree automata. In Agostino Cortesi and Gilberto Filé, editors, *Proceedings of the* 6th *International Static Analysis Symposium*, volume 1694 of *Lecture Notes in Computer Science*, pages 149–163, Venice, Italy, September 1999. Springer Verlag.

[28] Andrew Pitts. A note on logical relations between semantics and syntax. *Logic Journal of the IGLP*, 5(4):589–601, 1997.

[29] Gordon Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5(3):452–487, September 1976.

[30] M. Roe. *Cryptography and Evidence*. PhD thesis, University of Cambridge, 1997.

[31] Ian Stark. A fully abstract domain model for the π-calculus. In *Proceedings of the* 11th *Annual IEEE Symposium on Logic in Computer Science*, pages 36–42, New Brunswick, New Jersey, USA, July 1996. IEEE Computer Society.

[32] Roberto Zunino. Control flow analysis for the applied pi-calculus. *Electronic Notes in Theoretical Computer Science*, 99:87–110, August 2004.